Magnetic Levitation System

1EM

(MLS1EM)

Windows 7/10 x64

USB version

User's Manual

www.inteco.com.pl

Table of contents

1 Introduction

The *Magnetic Levitation System MLS* is a complete (after assembling and software installation) control laboratory system ready for experiments. The MLS is an ideal tool for demonstration of magnetic levitation phenomena. This is a classic control problem used in many practical applications like magnetic levitated trains. MLS uses both analogue and digital solutions to levitate a metallic ball (a sphere) due to an electromagnetic and gravitational forces.. *MLS* consists of the electro-magnet, the suspended hollow steel sphere, the sphere position sensors, computer interface board and drivers, a signal conditioning unit, connecting cables, real time control toolbox. This manual helps to teach students the automatic control applied to the MLS unstable, nonlinear and time varying dynamical system in real-time.

The basic principle of MLS operation is to apply the voltage to an electromagnet to keep a ferromagnetic sphere levitated. The sphere position is determined through a sensor. Additionally the coil current is measured to explore identification and multi loop or nonlinear control strategies. To levitate the sphere a real-time controller is required. The equilibrium stage of two forces (the gravitational and electro-magnetic) has to be maintained by this controller to keep the sphere in a desired distance from the magnet.

When two electromagnets are used then the lower one is used for external excitation or as contraction unit. This feature extends the MLS application range and is useful for robust controllers design.

The position of the sphere may be adjusted using the set-point control and the stability may be varied using the gain control. Two spheres of different diameter are provided. User-defined analogue controllers may be tested.

1.1 Laboratory set-up

A schematic diagram of the laboratory set-up is shown in Fig. 1.

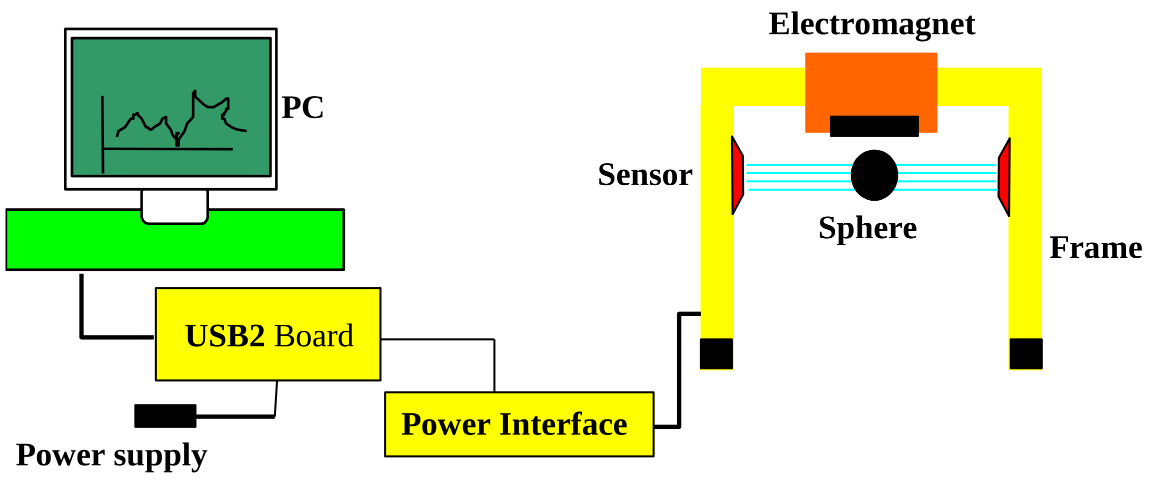


Fig. 1. MLS laboratory set-up

One obtains the mechanical unit with power supply and interface to a PC and the dedicated RTDAC/USB board configured in the Xilinx® technology. The software operates in real time under MS Windows® using MATLAB® and RTW (Simulink Coder) Toolbox.

Control experiments are programmed and executed in real-time in the MATLAB/Simulink environment. Thus it is strongly recommended to a user to be familiar with the RTW (Simulink Coder) toolbox. One has to know how to use the attached models and how to create his own models.

The control software for the MLS is included in the *MLS toolbox*. This toolbox uses the built in RT-CON from Inteco and Simulink and RTW (Simulink Coder) toolboxes from MATLAB.

MLS Toolbox is a collection of M-functions, MDL/SLX-models and C-code DLL-files that extends the MATLAB environment in order to solve MLS modelling, design and control problems. The integrated software supports all phases of a control system development:

- on-line process identification,
- control system modelling, design and simulation,
- real-time implementation of control algorithms.

MLS Toolbox is intended to provide a user with a variety of software tools enabling:

- on-line information flow between the process and the MATLAB environment,
- real-time control experiments using demo algorithms,
- development, simulation and application of user-defined control algorithms.

1.2 Hardware and software requirements.

MLS Toolbox is distributed on a CD-ROM or usb stick. It contains the software and *MLS User's Manual*. The *Installation Manual* is distributed in a printed form.

Hardware

Hardware installation is described in the *Installation Manual*. It consists of:

- Electromagnet
- Ferromagnetic spheres
- Position sensor
- Current sensor
- Power interface
- RTDAC/USB2 I/O board. The board contains FPGA equipped with dedicated logic,
- Pentium or AMD based personal computer.

Software

- Microsoft W7/W10x64 and MATLAB 64 bit with Simulink, and Simulink Coder (RTW) toolboxes (not included),
- Third party compiler MS Visual C++ depending on Matlab's version
- Details at: https://www.mathworks.com/support/sysreq/previous_releases.html
- The TCP/IP protocol must be installed in the computer system,
- CD-ROM or USB stick with MLS software and e-manuals (*User's Manual* and *Installation Manual*)

Details of the required software are available at:
http://www.inteco.com.pl/support/Software_requirements.pdf

⇨ **Real-time is supported by the RT-CON toolbox from INTECO (included in MLS Toolbox and transparent for a user).**

1.3 Features of MLS

One can highlight a number of features of MLS, among them the following deserve our attention

- The aluminum construction
- Two ferromagnetic objects (spheres) with different weights
- An optical detector to sense the object position
- A coil current sensor
- A highly nonlinear system ideal for illustrating complex control algorithms
- A frictionless system
- The full integration with MATLAB®/Simulink®
- Real-time control under MS Windows®

1.4 Typical teaching applications

- System Identification
- SISO, MISO, BIBO controllers design
- Intelligent/Adaptive Control
- Frequency analysis
- Nonlinear control
- Hardware-in-the-Loop
- Real-Time control
- Closed Loop PID Control

1.5 Software installation

Insert the installation CD or USB stick and proceed step by step following displayed commands.

2 MLS Main Window

The user has a rapid access to all basic functions of the MLS System from the *MLS Control Window*.

⇨ **If the MATLAB R2018 or newer is used run the *rehash toolbox* command, close Matlab and open again.**

Then type:

mls1em_usb2_main

and then the *Magnetic Levitation Main* window opens (see Fig. 2).

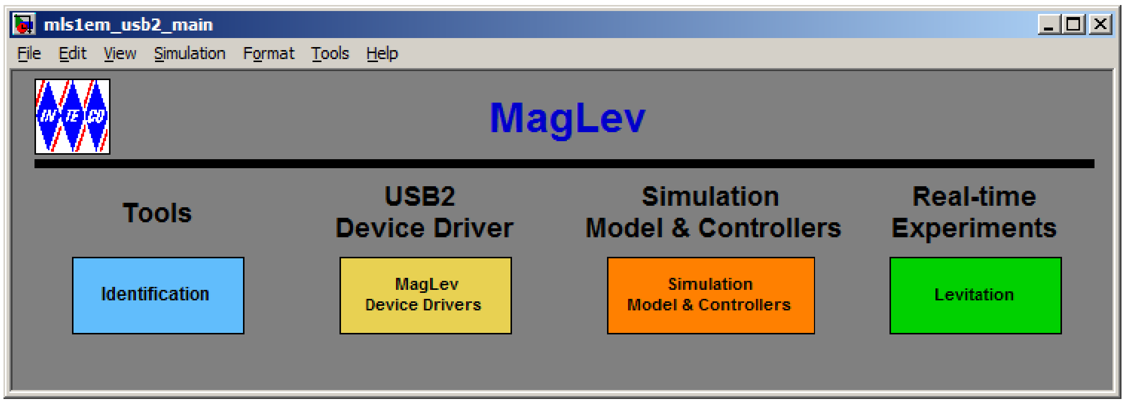


Fig. 2. The Magnetic Levitation Main window

In the ML Main window one can find: testing tools, drivers, models and demo applications. You can see a number of pushbuttons ready to use.

The *ML Main* window shown in Fig. 2 contains four buttons:

- Tools – identification
- USB2 Device Driver – MagLev device driver
- Simulation model and controllers
- Real-time experiments – levitation

Section 2 is divided into four subsections. Under each button in the ML Main window one can find the respective portion of software corresponding to the problem announced by the button name. These problems are described below in four consecutive subsections.

2.1 Identification

If we click the identification button the following window (see Fig. 3) opens. There are the default values of all parameters defined by the manufacturer. Nevertheless, a user is equipped with a number of identification tools. He can perform the identification procedures to verify and if necessary modify static and dynamic characteristics of MLS.

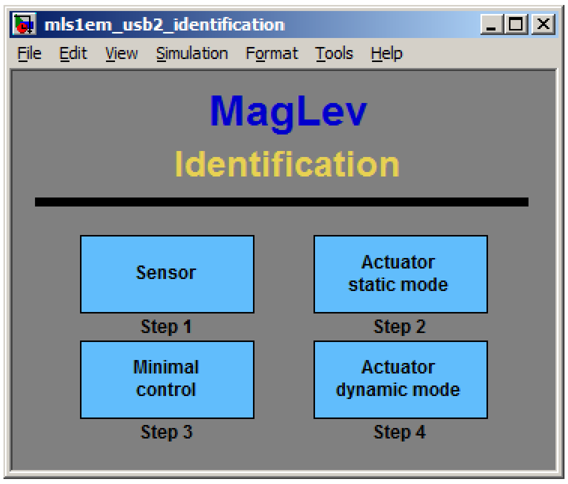


Fig. 3. The identification window

Four identification steps have been preprogrammed. They are described below.

A special current driver has been built into the power interface of MLS. In this way, the control (the coil current signal) has been customized to the very short sampling time (0.6 ms) despite the fact that the USB transmission protocol requires a longer sampling time.

2.1.1 Sensor

In this subsection the position sensor characteristics is identified.

If you click the Sensor button the following window opens (see Fig. 4)

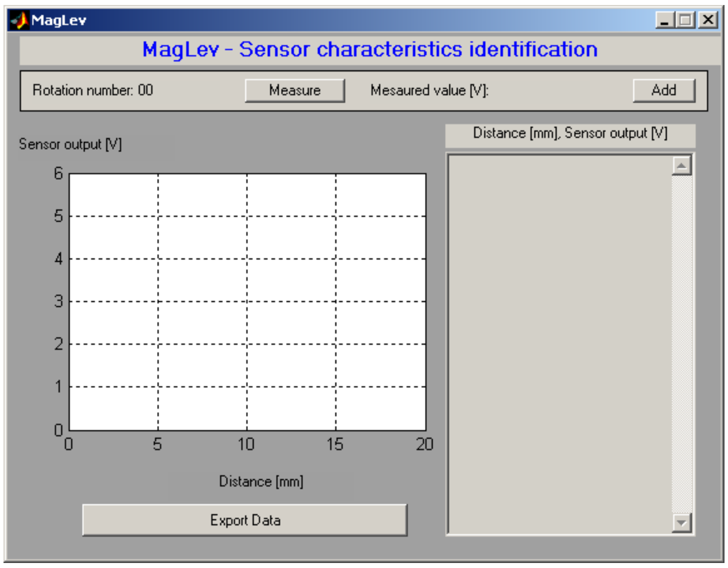
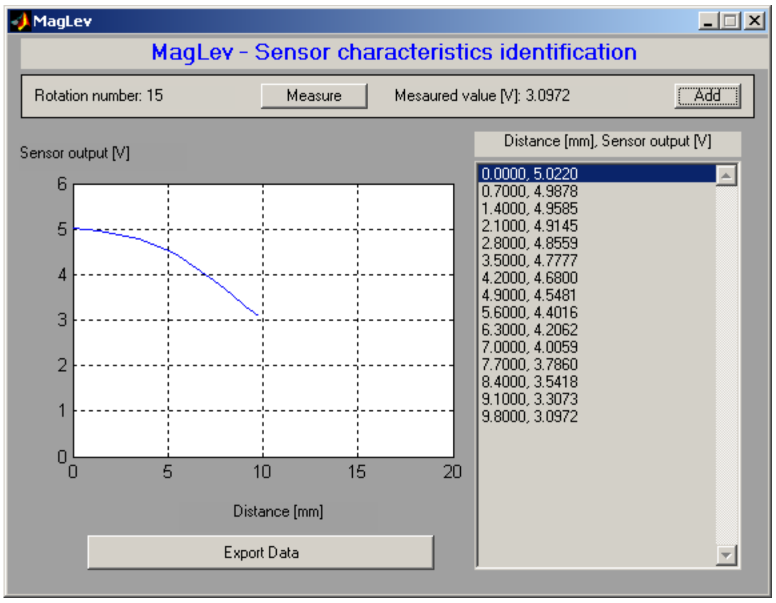


Fig. 4. Sensor signal in [V] vs. the sphere distance from the electromagnet in [mm]

The following procedure is required to identify the characteristics.

1. Screw in the screw bolt into the seat.
2. Screw in the black sphere and lock it by the butterfly nut. **Notice that the sphere is fixed to the frame!**
3. Turn round the screw so the sphere is in touch with the bottom of the electromagnet.
4. Switch on the power supply and the light source.
5. Start the measuring and registration procedure. It consists of the following steps:
6. Push the *Measure* button – the voltage from the position sensor is stored and displayed as *Measured value [V]*. One can correct this value by measuring it again.
   - Push the Add button – the measured value is added to the list. A rotation number value is automatically enlarged by one (see Fig. 5).



Rotation number: 15

Mesaured value [V]: 3.0972

Distance [mm], Sensor output [V]

```
0.0000, 5.0220
0.7000, 4.9878
1.4000, 4.9585
2.1000, 4.9145
2.8000, 4.8559
3.5000, 4.7777
4.2000, 4.6800
4.9000, 4.5481
5.6000, 4.4016
6.3000, 4.2062
7.0000, 4.0059
7.7000, 3.7860
8.4000, 3.5418
9.1000, 3.3073
9.8000, 3.0972
```

Fig. 5. Characteristic of the sphere position sensor

- Manually make one full rotation of the screw.
- Repeat three last steps several times until no change in the voltage vs. position is observed.

7. Push the *Export Data* button – the data are written to the disc. Data are stored in the ***mls1em_usb2_sensor.mat*** file as the *SensorData* structure with the following signals: *Distance_mm*, *Distance_m* and *Sensor_V*.

In the Simulink real-time models the above characteristics is used as a Look-Up-Table model. The block named *Position scaling* is located inside the device driver block of MLS (see Fig. 7). Notice, that the characteristic shows meters vs. Volts. In Fig. 6 there were shown Volts vs. meters. It is obvious that we require the inverse characteristics because we need to define the output as the position in meters.

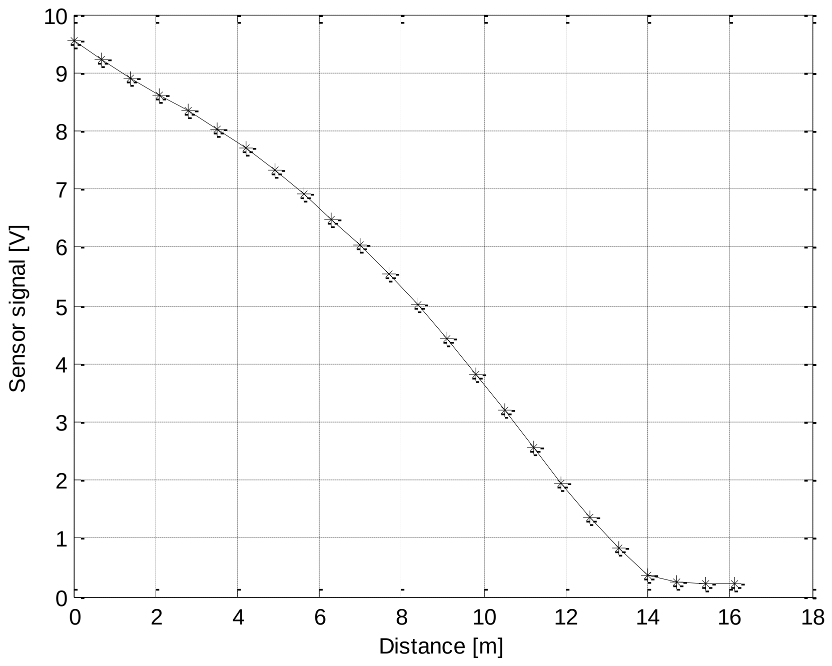


Fig. 6. The sensor characteristic after being measured and exported to the disc

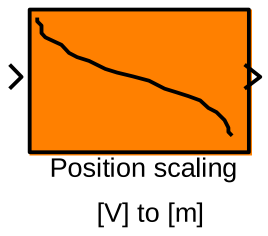


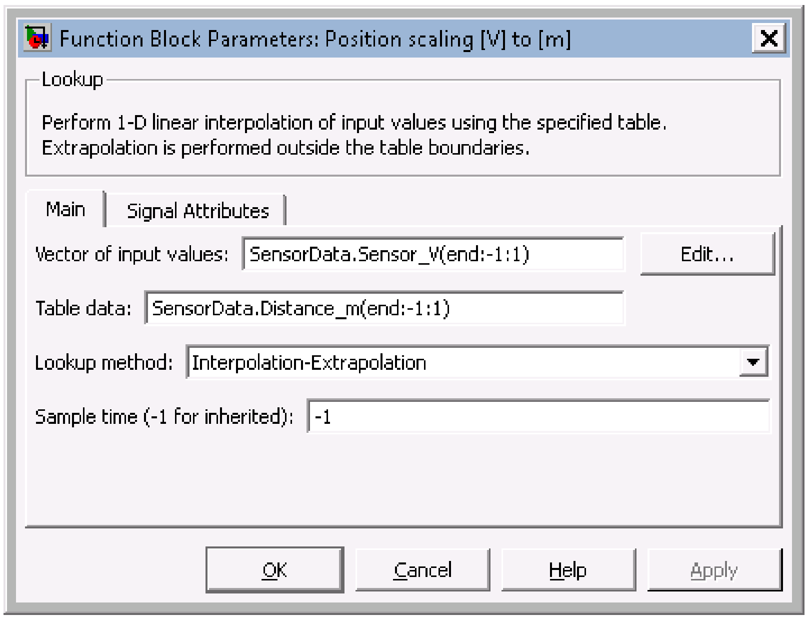
Fig. 7. The Simulink Look Up Table model representing the position sensor characteristics

If we click this block the window shown in Fig. 8 opens. Any time you like to modify the sensor characteristics you can introduce new data related to the voltage measured by the sensor. The voltage corresponds to the distance of the sphere set by a user while the identification procedure is performed. The sensor characteristic is loaded from the ***mls1em_usb2_sensor.mat*** file which has been created during the identification procedure. If the curve of the *Position scaling* block is not visible please load the file with data.

The sensor characteristics can be approximated by a polynomial of a given order. For example, we can use a fifth order polynomial.

$$P(x) = p_5 x^5 + \ldots + p_0$$

$p_5 =$ 0.000015564899124, $p_4 =$ -0.000113896440838, $p_3 =$ -0.004250246343072, $p_2 =$ 0.026751358403873, $p_1 =$ -0.475668807418930 and $p_0 =$ 9.541344731311472.


Function Block Parameters: Position scaling [V] to [m]

Lookup

Perform 1-D linear interpolation of input values using the specified table.
Extrapolation is performed outside the table boundaries.

Vector of input values: SensorData.Sensor_V(end:-1:1)

Table data: SensorData.Distance_m(end:-1:1)

Lookup method: Interpolation-Extrapolation

Sample time (-1 for inherited): -1

Fig. 8. Look-Up Table to be fulfilled with vectors of input and output values

The approximated polynomial (the red line) is shown in Fig. 9. The polynomial approximation will be not used in this manual due to the fact that the entire model is built in Simulink. Therefore we recommend to model the characteristics as a Look-Up Table block (see Fig. 7 and Fig. 8).

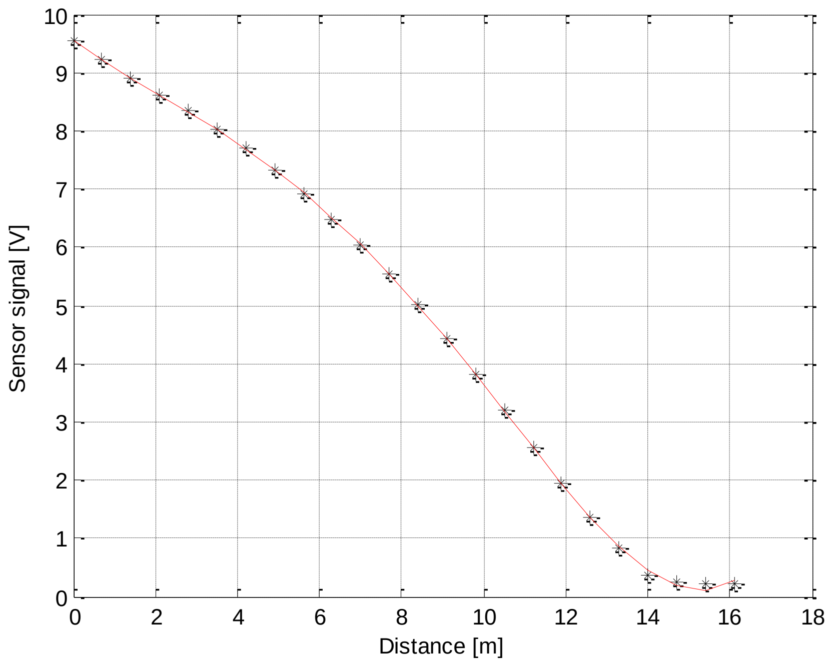


Fig. 9. The sensor characteristic approximated by the fifth order polynomial

Please note that the sensor characteristics can vary. For a different sphere the sensor characteristics is also different. The identification is required when performing an experiment with a new sphere.

2.1.2 Actuator static mode

In this subsection we examine static features of the actuator i.e. the electromagnet. Notice, that **the sphere is not present**!

Click the *Actuator static mode* button and the window shown in Fig. 10 opens.

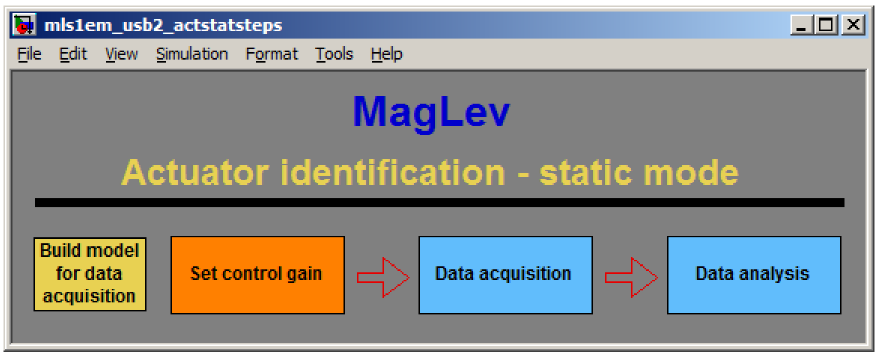

mls1em_usb2_actstatsteps
File Edit View Simulation Format Tools Help
MagLev
Actuator identification - static mode
Build model for data acquisition
Set control gain
Data acquisition
Data analysis

Fig. 10. Identification window of a static current/voltage characteristic

Now, we can perform button by button the operations depicted in Fig. 10. We begin from the *Build model for data acquisition* button. The window of the real-time task shown in Fig. 11 opens and the RTW build command is executed (the executable code is created).

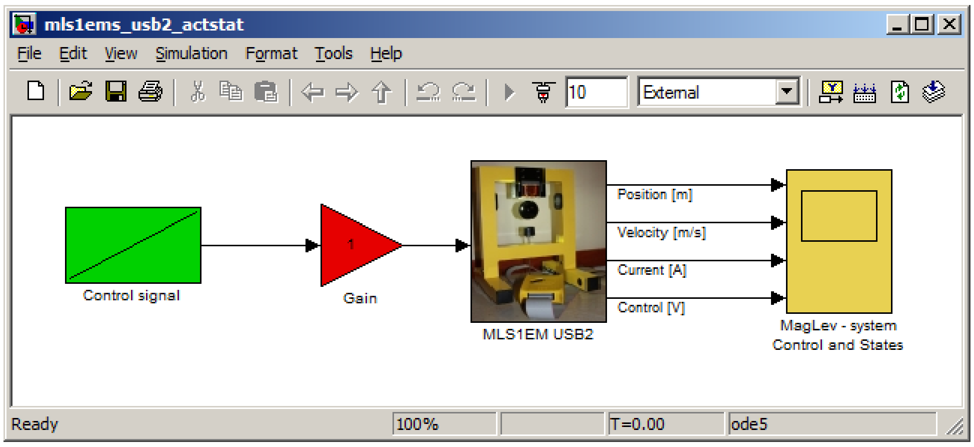
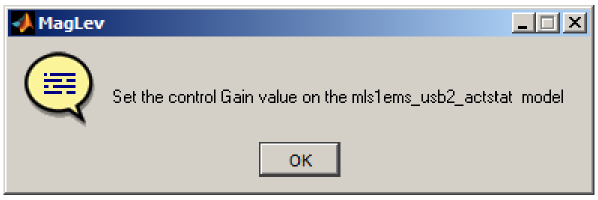


Fig. 11. Real-time model built to examine the current in the electromagnetic coil

Click the *Set control gain* button. It results in activation of the model window and the following message is displayed (see Fig. 12):



Set the control Gain value on the mls1ems_usb2_actstat model

Fig. 12. Message – Set the "Control Gain"

In Fig. 11 one can notice the *Control signal* block. In fact the control signal increases linearly. We can modify the slope of this signal changing the *Control Gain* value.

Click the *Data acquisition* button. Within 10 seconds data are acquired and stored in the workspace.

Click the *Data analysis* button. The collected values of the coil current are displayed in Fig. 13.

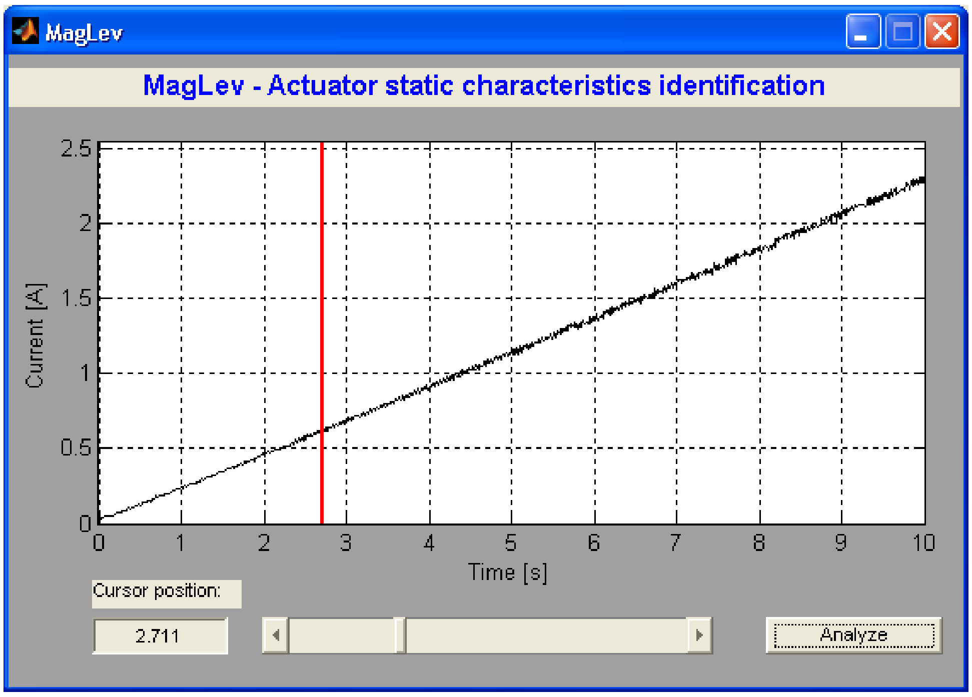
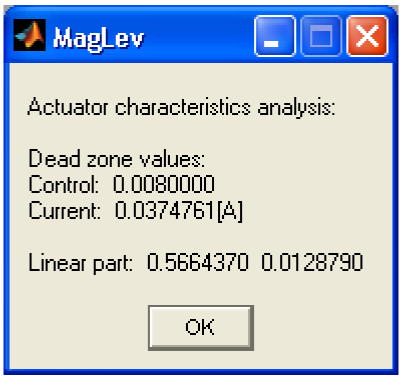


Fig. 13. Current in the electromagnetic coil

The characteristic is linear. We can locate the cursor at the point where a new line slope starts (see the red line in the picture). We can move the cursor in two ways: by writing down a value into the edition window or by drugging the slider. In this way the current characteristics is prepared to be analyzed in the next step. The line is divided into two intervals: the first – from the beginning of measurements to the cursor and the second – from the cursor to the end of measurements. If the dead zone is observed – locate the cursor at the and of it. The left part is considered as constant and the right as linear.

After setting the cursor position, consequently, click the *Analyze* button. The following message (see Fig. 14) appears. The constants *a* and *b* of the linear part are the parameters of the line equation: $i(u) = a\,u + b$.



Actuator characteristics analysis:

Dead zone values:
Control: 0.0080000
Current: 0.0374761[A]

Linear part: 0.5664370 0.0128790

Fig. 14. Coefficients of the actuator characteristic

These parameters, namely: $u_{MIN} = 0.0$, $u_{MAX} = 5.0$ $x_{3MIN} = 0.0345$, $x_{3MAX} = 2.345$, are going to be used in the simulation model in section 2.3.1 (see the differential equations parameters).

To obtain a family of static characteristics for linear controls with different slopes we repeat the following experiment. We apply a control voltage signal in the time interval from 0 to 10 s. The control signal in the range of [0,1] was multiplied by the user defined gain. Fig. 15 shows the current response for gain set to 1, 3 and 5.

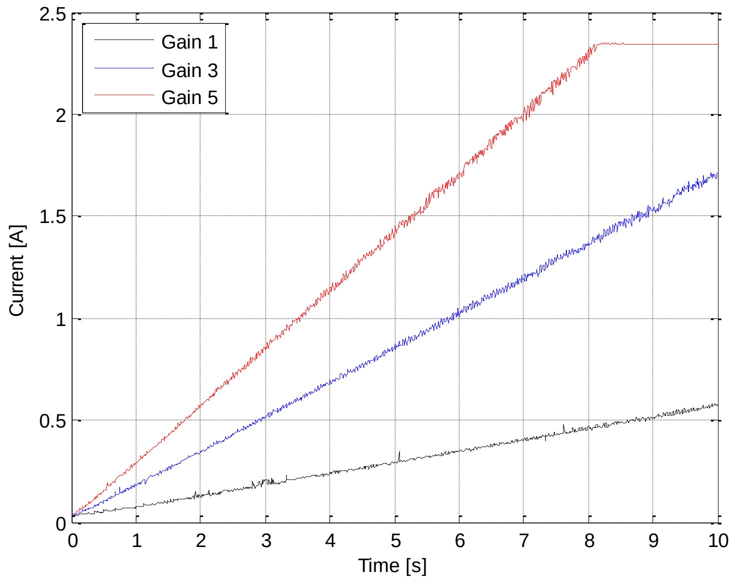


Fig. 15. Family of the output (current) characteristics

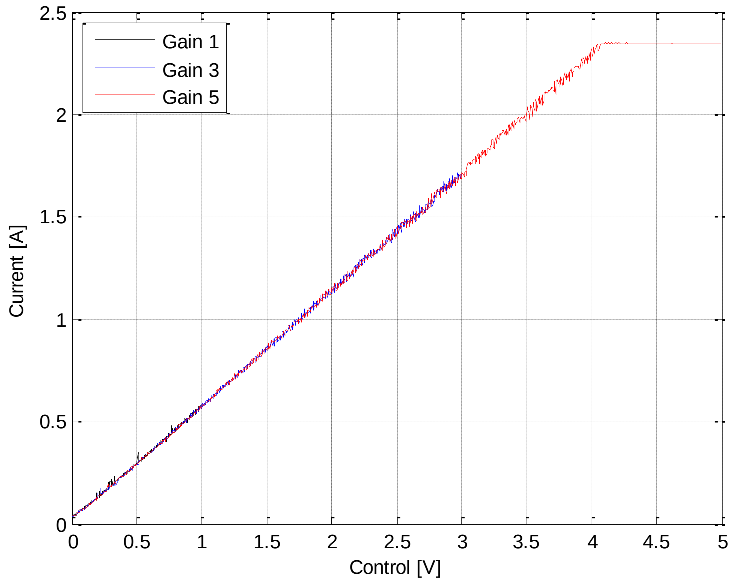


Fig. 16. Current vs. Control voltage

Consequently, we obtain diagrams of the currents corresponding to ten experiment (see Fig. 15). Each characteristic is approximated by a polynomial of the first order. Finally the entire current vs. control relation is depicted in Fig. 16. The red line represents the linear approximation of measurements. We obtain the following numerical values of linear characteristics: a = 0.5447, b = 0.0233.

2.1.3 Minimal control

In this subsection we examine the minimal control to cause a forced motion of the sphere from the supporting structure (tablet) toward the electromagnet against the gravity force. Notice, that in this experiment **the sphere is not levitating**! It is kept nearby the electromagnet by the supporting structure.

Click the Minimal control button and the window shown in Fig. 17 opens.

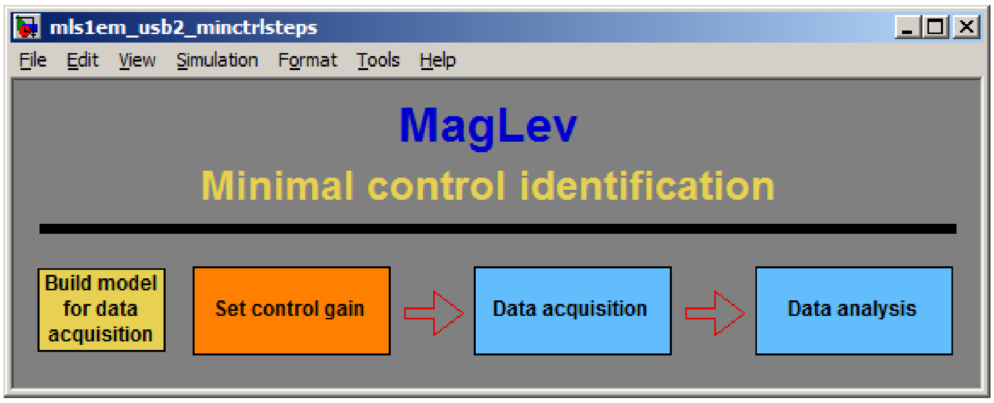

mls1em_usb2_minctrlsteps
File Edit View Simulation Format Tools Help
MagLev
Minimal control identification
Build model for data acquisition
Set control gain
Data acquisition
Data analysis

Fig. 17. Window to identify the minimal control vs. distance (between the sphere and electromagnet)

Now, we proceed button by button the operations depicted in Fig. 17 similarly to the procedure described in the previous subsection. We begin from the *Build model for data acquisition* button. The window of the real-time task shown in Fig. 18 opens.

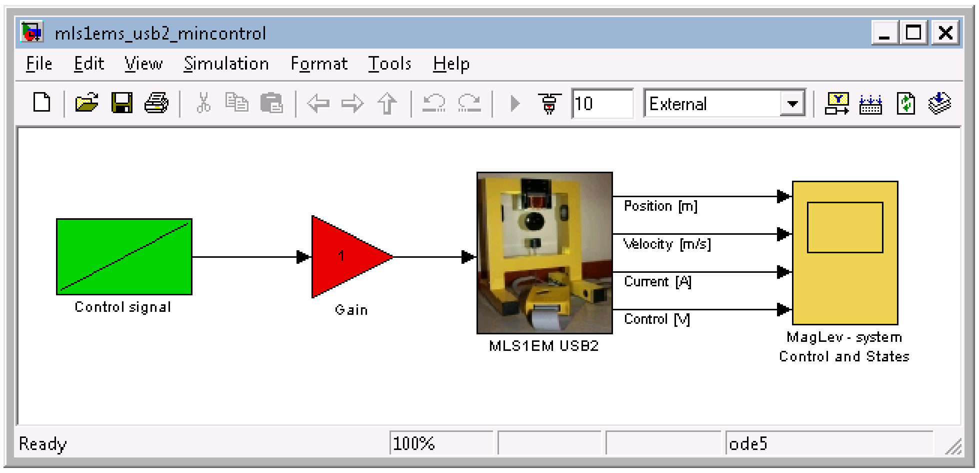
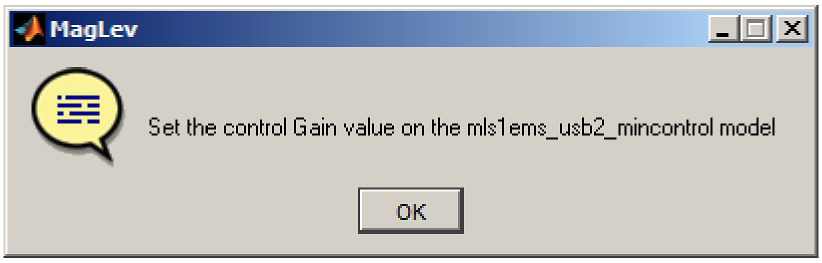


Fig. 18. Real-time model built to examine the minimal electromagnetic force

Click the *Set control gain* button. This results in the activation of the model window and the following message is displayed (see Fig. 19).



Set the control Gain value on the mls1ems_usb2_mincontrol model

Fig. 19. Message – set the "Control Gain"

It means that we can set a duty cycle of the control PWM signal. The sphere is located on the support and the experiment starts. Click the *Data acquisition* button. A forced motion of the ball toward the electromagnet begins.

Click the *Data analysis* button. The collected values of the ball position are displayed in Fig. 20.

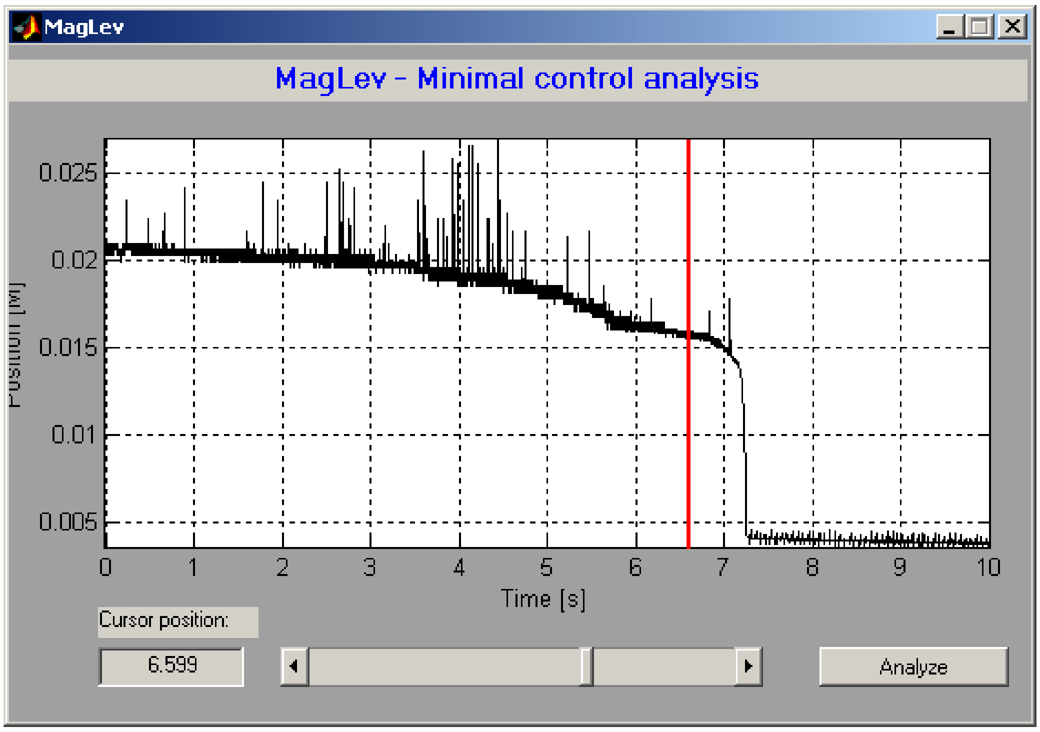
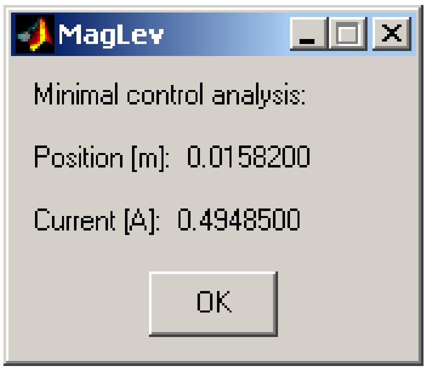


Fig. 20. The sphere motion

The sphere motion is visible. We can locate the cursor at the point slightly before a position jump occurs (takes place) (see the red line in the picture). We can move the cursor in two ways: by writing down a value into the edition window or by drugging the slider. In this way the acquired data are prepared to be analyzed in the next step.

After setting the cursor position, consequently, click the *Analyze* button. The following message (see Fig. 21) appears. This information means that the sphere located 15.82 mm from the electromagnet begins to move toward it when the PWM control over-crosses the 0.49485 duty cycle value.


MagLev

Minimal control analysis:

Position [m]: 0.0158200

Current [A]: 0.4948500

Fig. 21. Message of the experiment results

2.1.4 Actuator dynamic mode

In this subsection we examine dynamic features of the actuator i.e. the electromagnet. It means that the moving sphere generates an electromotive force (EMF). EMF diminishes the current in the electromagnet coil. The hardware coil-current driver is used to minimize this effect. The dynamical actuator (the electromagnet and current driver hardware electronics) must be identified to find a time constant of the actuator. To achieve this the step type actuator excitation is used.

Click the Actuator static mode button and the window shown in Fig. 22 opens.

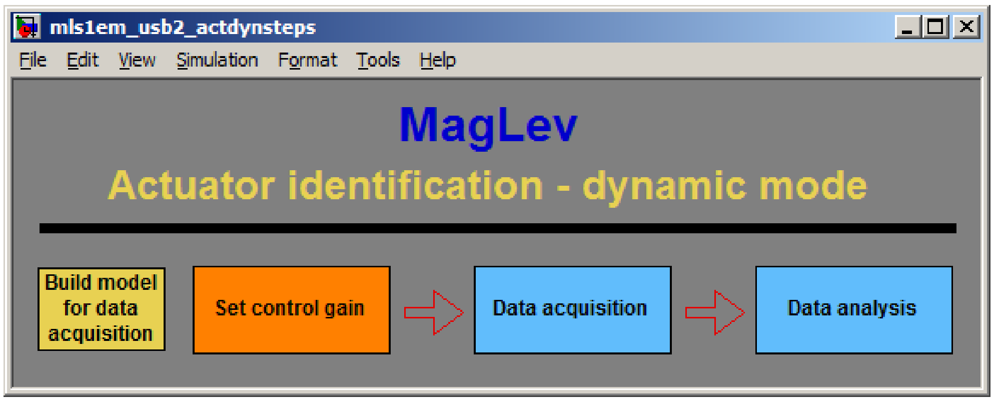

mls1em_usb2_actdynsteps
File Edit View Simulation Format Tools Help
MagLev
Actuator identification - dynamic mode
Build model for data acquisition
Set control gain
Data acquisition
Data analysis

Fig. 22. Identification window of a dynamic current/voltage characteristic

A user should perform three experiments: without the sphere (Without ball), with the sphere on the supported structure (Ball on the tablet) and with the sphere fixed to the rigid screw (Ball fixed). We begin from the Build model for data acquisition button. The window of the real-time task shown in Fig. 23 opens. We have to set the control gain. If we are going to modify the control magnitude then we set the default gain to 1 and the subsequent duty cycles to: 0.25, 0.5, 0.75 and 1. Click the Data acquisition button and save data under a given file name.

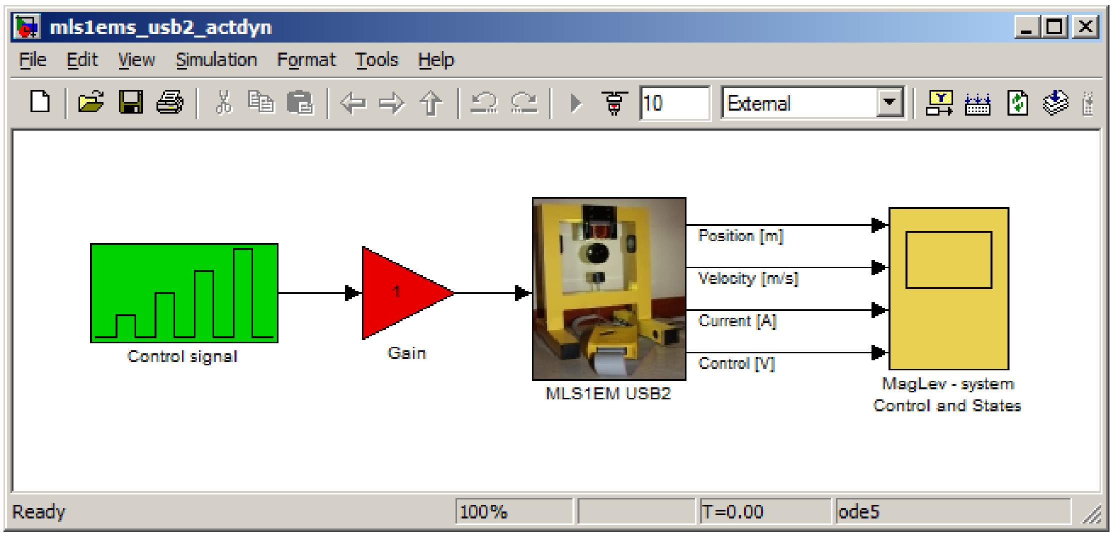
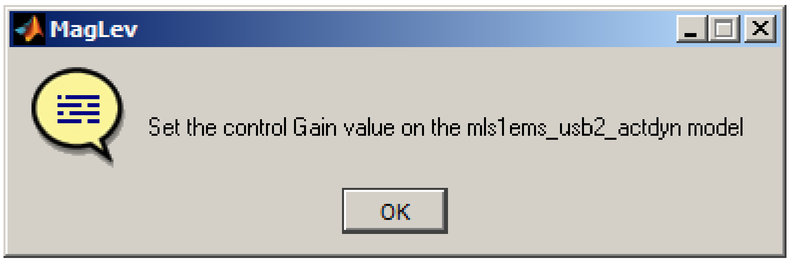


Fig. 23. Real-time model built to examine the EMF influence on the coil current

Click the *Set control gain* button. It results in activation of the model window and the following message is displayed (see Fig. 24).



Set the control Gain value on the mls1ems_usb2_actdyn model

Fig. 24. Message – set the “Control Gain”

Click the *Data analysis* button. It calls the *mls1em_usb2_find_curr_dyn.m* file. The parameters optimization procedure starts. The optimization routine is based on the *mls1emm_usb2_current.mdl* model.

When mls1em_usb2_find_curr_dyn.m runs the optimization function fminsearch is executed. Fminsearch uses the mls1em_opt_current.m file.

The k_i and f_i parameters are iteratively changed during the optimization procedure. The current curve is fitted four times. This is due to the control signal form.

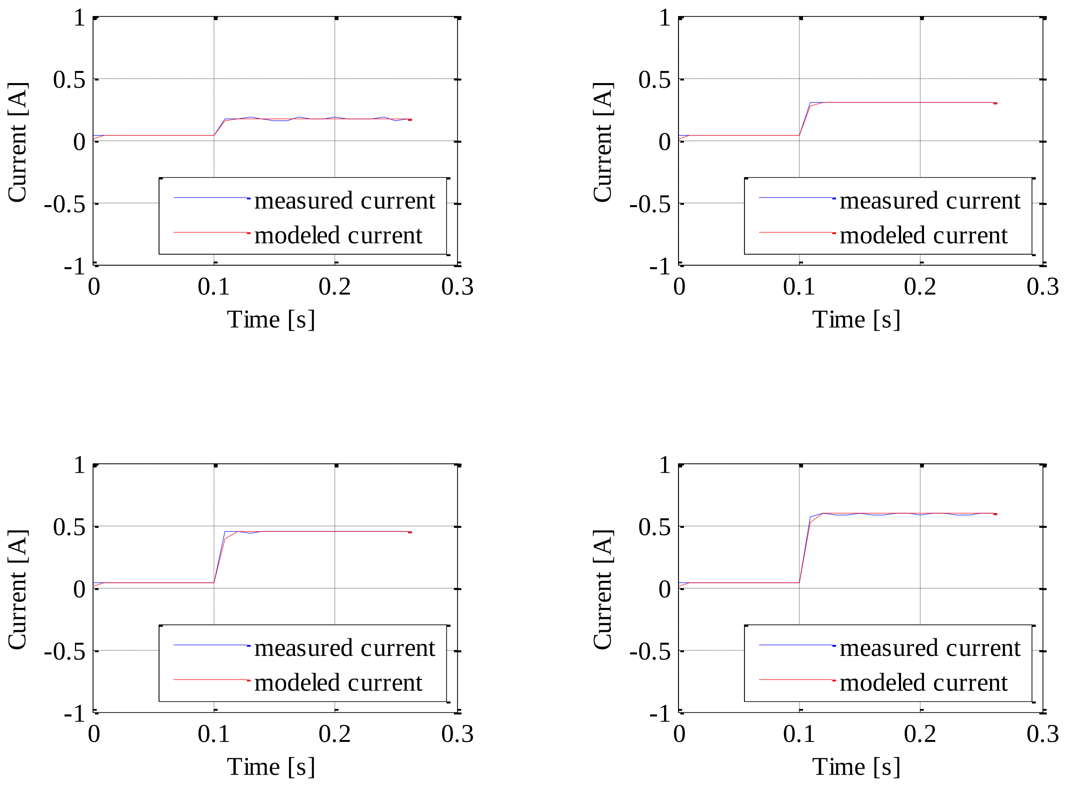
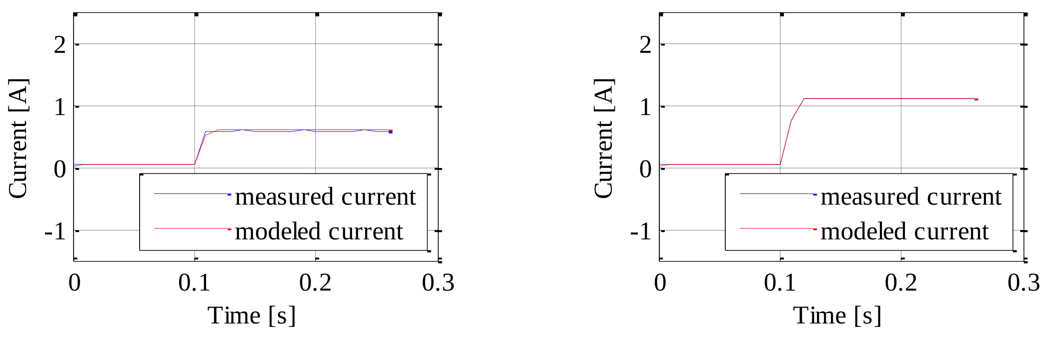
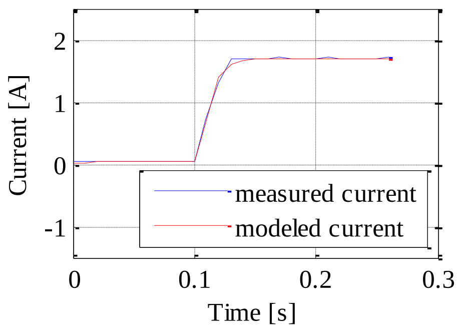


Fig. 25. Measured and modeled coil current for every of the four step excitations. Gain 1.





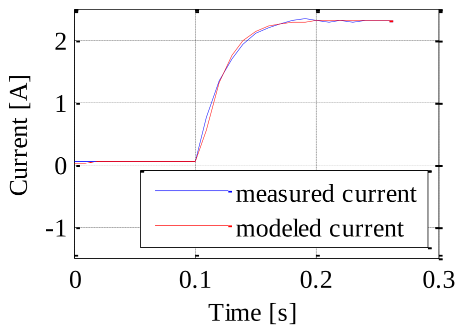
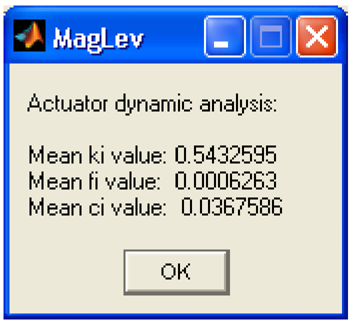


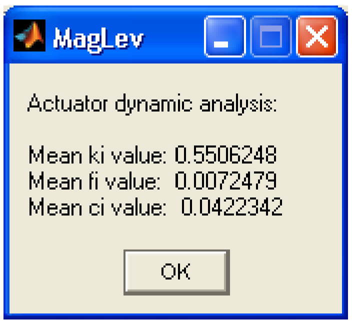
Fig. 26. Measured and modeled coil current for every of the four step excitations. Gain 4.

Finally the information about the mean values is displayed (see Fig. 27). The advanced user can use the functions code to perform a detailed analysis.



Actuator dynamic analysis:

Mean ki value: 0.5432595
Mean fi value: 0.0006263
Mean ci value: 0.0367586



Actuator dynamic analysis:

Mean ki value: 0.5506248
Mean fi value: 0.0072479
Mean ci value: 0.0422342

Fig. 27. Optimisation results, a) Gain 1, b) Gain 4

	Gain 1			Gain 4		
	k_i	f_i	c_i	k_i	f_i	c_i
step 1	0.5353	$0.6267 \cdot 10^{-3}$	0.03630	0.5531	$0.6272 \cdot 10^{-3}$	0.03824
step 2	0.5368	$0.6261 \cdot 10^{-3}$	0.03694	0.5332	$2.7961 \cdot 10^{-3}$	0.03568
step 3	0.5445	$0.6265 \cdot 10^{-3}$	0.03741	0.5540	$8.3108 \cdot 10^{-3}$	0.03859
step 4	0.5564	$0.6257 \cdot 10^{-3}$	0.03636	0.5623	$17.2575 \cdot 10^{-3}$	0.05642

One can find that the control gain - *ki* parameter is constant for every experiment, but the time constant *fi* varies for a large control signal change. It means, that depending on the planed control action it is recommended to use the appropriate model. In the research literature one can find simplified MagLev models when the time constant is neglected. It is justified assumption for a stabilization task. For example when the time-optimal control is used and the bang-bang type control signal is applied, the model must be supplemented with the time constant valid for the largest change of the control signal.

Analyzing the optimized parameters values, one can find that the time constant is very small – about 0.62 ms for control signals used for the stabilization mode. For the maximal change in the control signal the time constant increases up to 17 ms. The time constant dependency vs. control gradient can be approximated by the second order polynomial (See Fig. 28).

Please note, that in the case of MLS control via USB board the sampling time is set to 10 ms.

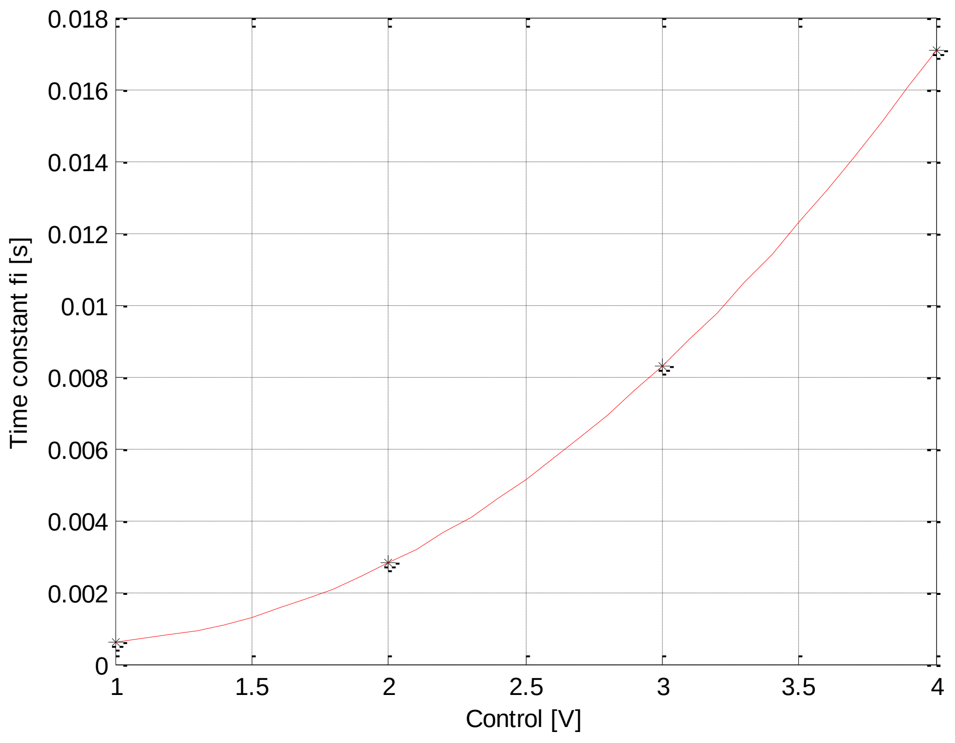


Fig. 28. Actuator time constant vs. control gradient

Finally, for the mathematical model the time constant f_i = 0.0006263 is used.

2.2 MagLev device drivers

The driver is a software go-between for the real-time MATLAB environment and the RT-DAC/USB external module. The control and measurements are driven. Click the *MagLev Device Drivers* button in the *Magnetic Levitation Main* window. The following window opens (see Fig. 29).

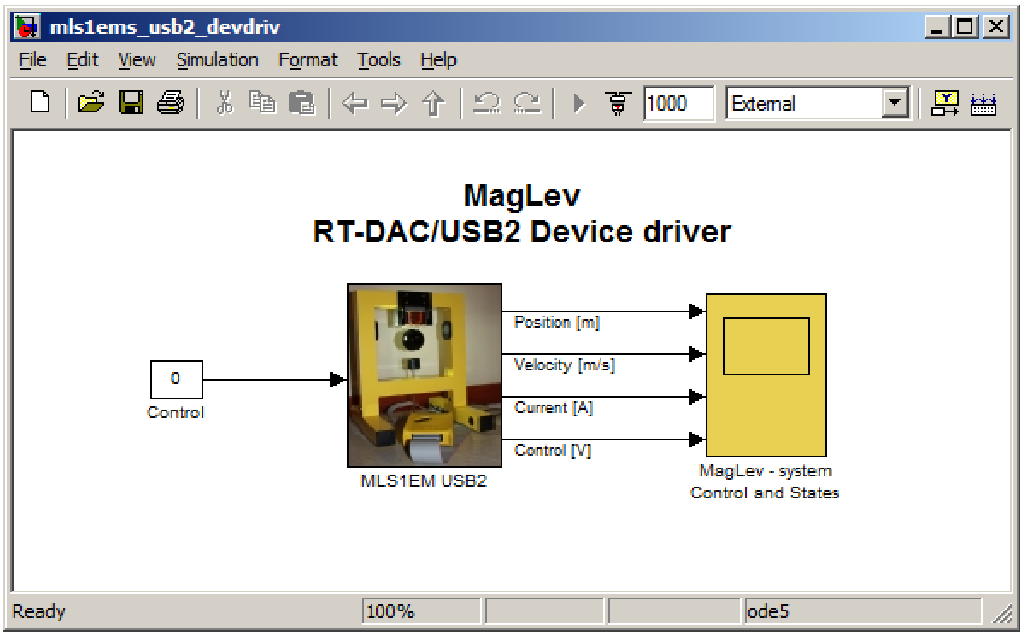


Fig. 29. USB MagLev device driver window

Notice that the scope block writes data to the *MLExpData* variable defined as a structure with time. The structure consists of the following signals: Position [m], Velocity [m/s], Current [A], Control [V]. The interior of the *MLS1EM USB2* block, it means the interior of the driver block is shown in Fig. 30.

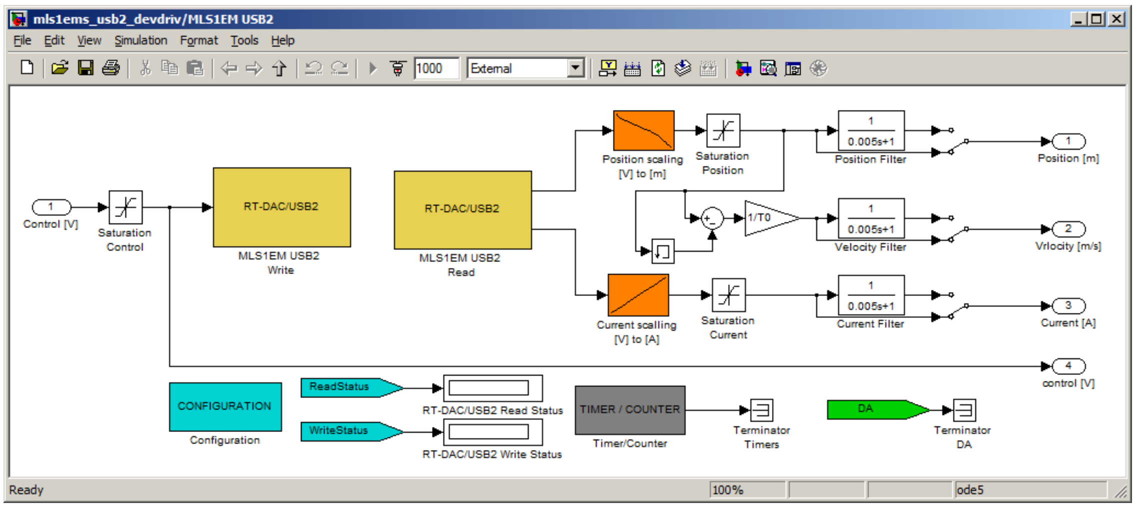


Fig. 30. Interior of the driver block

In fact there are two drivers: *MLS1EM USB2 Read* and *MLS1EM USB2 Write*. There are also two characteristics: the ball position [m] vs. the position sensor voltage [V] and the coil current vs. the current sensor voltage [V]. The driver uses functions which communicate directly with a logic stored at the RT-DAC/USB2 module. When one wants to build his own application one can copy this driver to a new model. The interior of the device drivers block is presented in Fig. 31 and Fig. 32. The information between the MATLAB/Simulink and RT-DAC/USB2 board is exchanged by the driver files *mls1em_usb2_ddr* and *mls1em_usb2_ddw* devoted to read and write operations via USB.

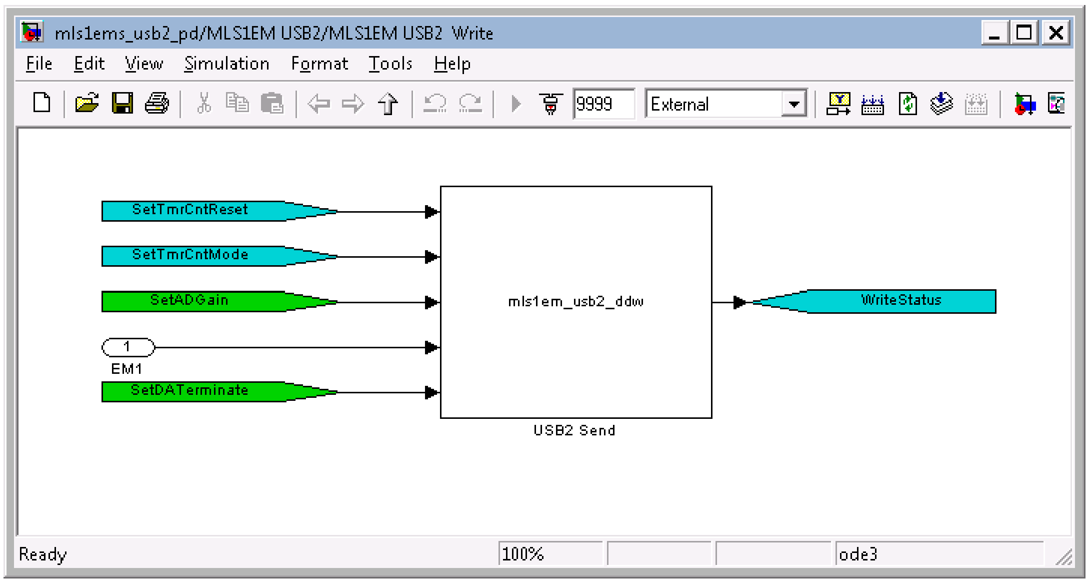


Fig. 31. USB Send device driver file connections

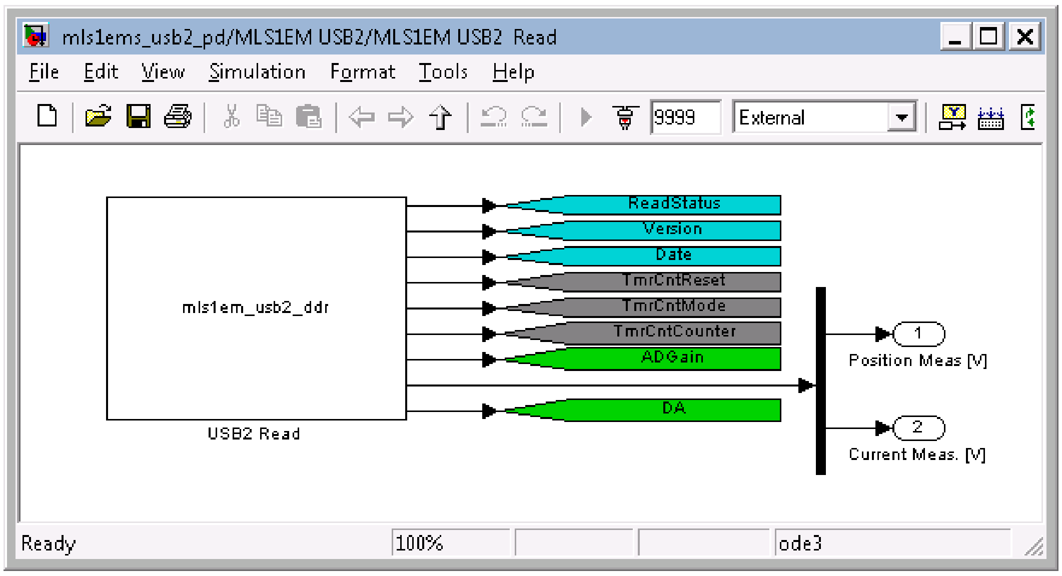


Fig. 32. USB Read device driver file connections

The *Configuration* block allows receiving information about FPGA logic and set the A/D Gain as well terminate the A/D conversion (see Fig Fig. 33).

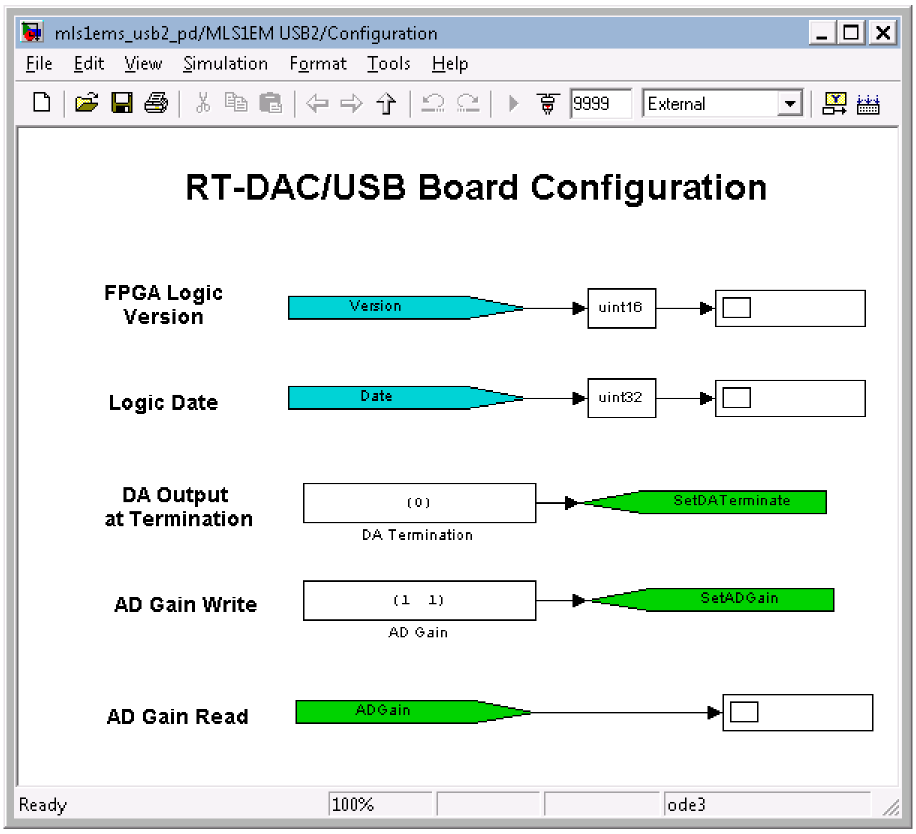


Fig. 33. RT-DAC/USB module configuration block

The Timer Counter block allows configuring the FPGA Timers/Counters. Fig. 34 presents the Timer configuration. The received value is displayed and returned from the block.

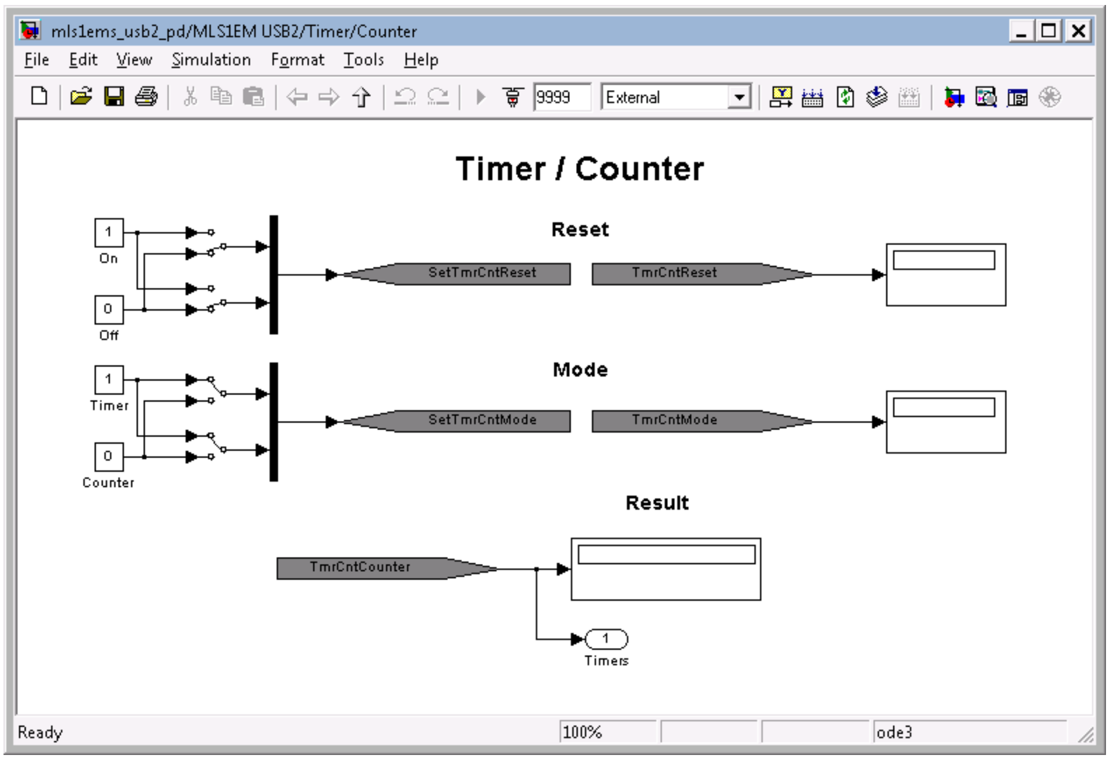


Fig. 34. RT-DAC/USB Timer/Counter configuration block

⇨ **Do not introduce any changes inside the original driver. They can be introduced only inside its copy!!! Make a copy of the installation CD.**

The Simulink Look-Up-Table model named *Position scaling* (see Fig. 7) representing the position sensor characteristics has been already described. Now let us present the second Simulink Look-Up-Table model named *Current scaling* (see Fig. 35).

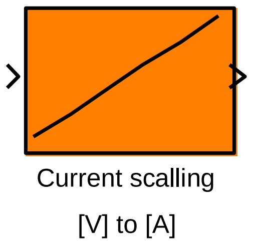


Fig. 35. The Simulink Look-Up-Table model representing the current sensor characteristic

To build the above characteristic it is necessary to measure the current of the electromagnet coil. The algorithm in the computer is the source of the desired value of the **control in the form of the voltage signal. This characteristic has been built by the manufacturer. It** is not recommended to repeat measurements by a user because to do so one must unsolder the input wires of the electromagnet.

2.3 Simulation Model & Controllers

Click the *Simulation Model & Controllers* button in the *Magnetic Levitation Main* window. The following window opens (see Fig. 36).

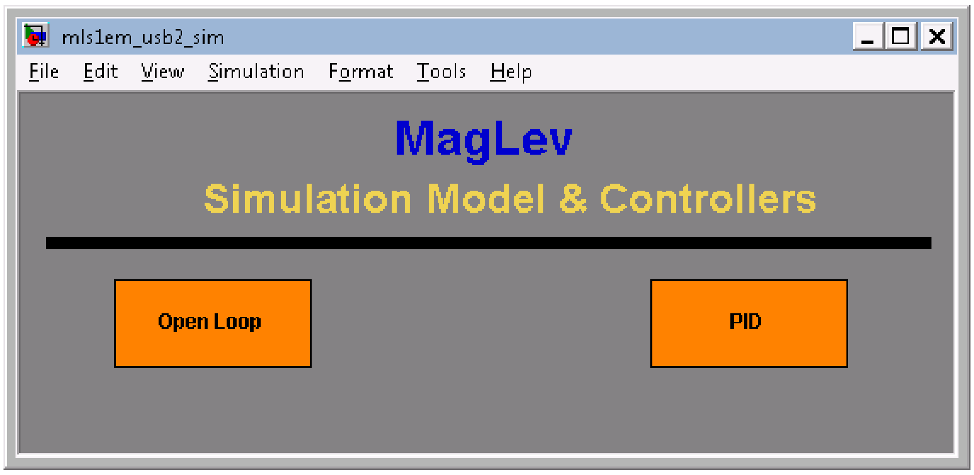

mls1em_usb2_sim
File Edit View Simulation Format Tools Help
MagLev
Simulation Model & Controllers
Open Loop
PID

Fig. 36. Simulation Model & Controllers window

2.3.1 Open Loop

- **Simulink model**

Next, you can click the first *Open Loop* button. The following window opens (see Fig. 37). Notice that the scope block writes data to the *MLSimData* variable defined as a structure with time. The structure consists of the following signals: Position [m], Velocity [m/s], Current [A], Control [V].

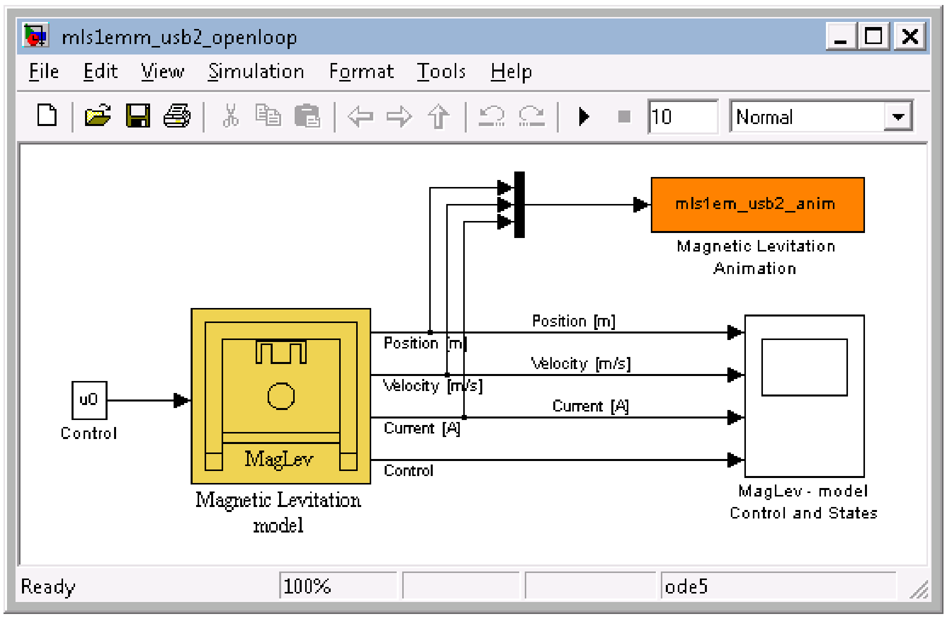
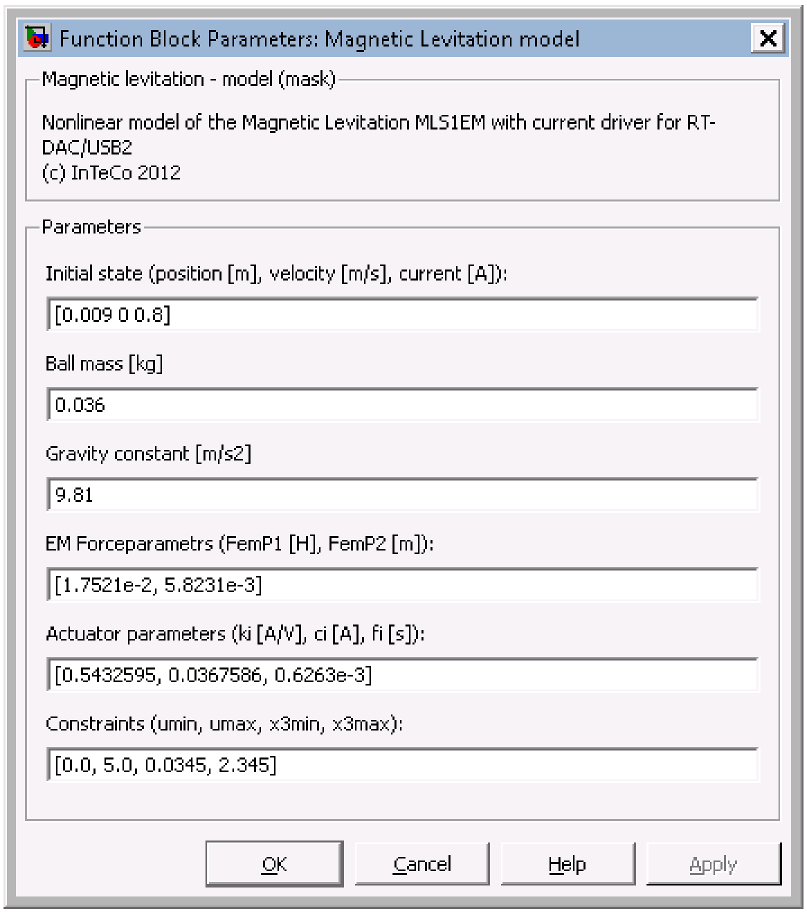

mls1emm_usb2_openloop
File Edit View Simulation Format Tools Help
10
Normal
mls1em_usb2_anim
Magnetic Levitation Animation
Position [m]
Velocity [m/s]
Current [A]
Control
u0
Control
MagLev
Magnetic Levitation model
MagLev - model Control and States
Ready
100%
ode5

Fig. 37. Open-loop simulation

If you click the *Magnetic Levitation model* block the following mask opens (see Fig. 38).



Function Block Parameters: Magnetic Levitation model

Magnetic levitation - model (mask)

Nonlinear model of the Magnetic Levitation MLS1EM with current driver for RT-DAC/USB2
(c) InTeCo 2012

Parameters

Initial state (position [m], velocity [m/s], current [A]):
[0.009 0 0.8]

Ball mass [kg]
0.036

Gravity constant [m/s2]
9.81

EM Forceparametrs (FemP1 [H], FemP2 [m]):
[1.7521e-2, 5.8231e-3]

Actuator parameters (ki [A/V], ci [A], fi [s]):
[0.5432595, 0.0367586, 0.6263e-3]

Constraints (umin, umax, x3min, x3max):
[0.0, 5.0, 0.0345, 2.345]

Fig. 38. Mask of the Magnetic Levitation model

In Fig. 37 go to the *File* option and choose *Look under mask*. The interior of the *Magnetic Levitation model* block shown in Fig. 39 opens.

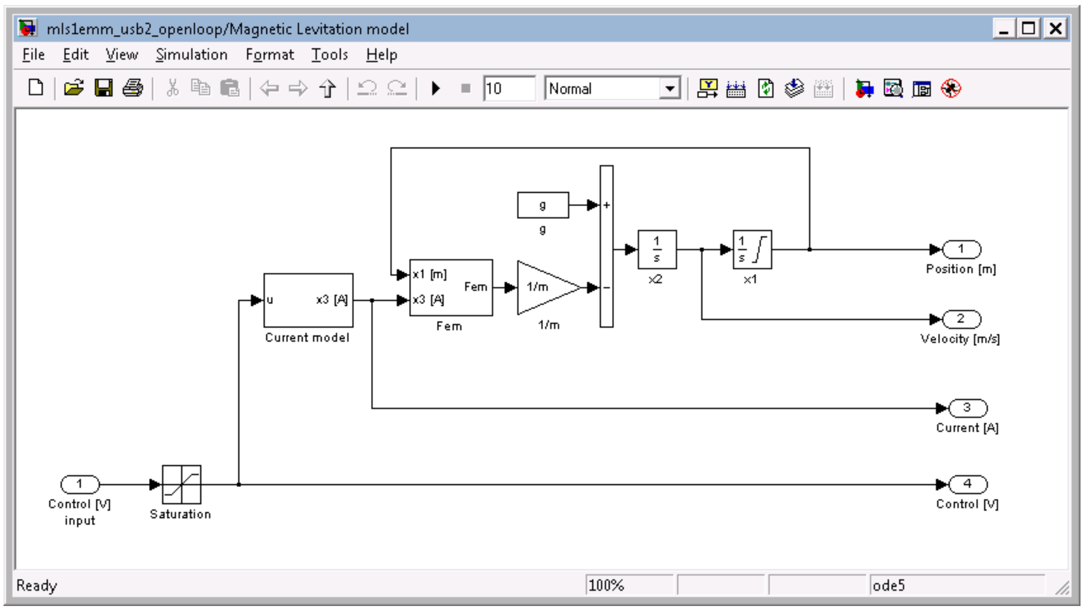


Fig. 39. Interior of the ML model

Notice that there are two integrator blocks in Fig. 39. In fact we deal with third order dynamical system. The third integrator relates to the coil current. This is visible in Fig. 40.

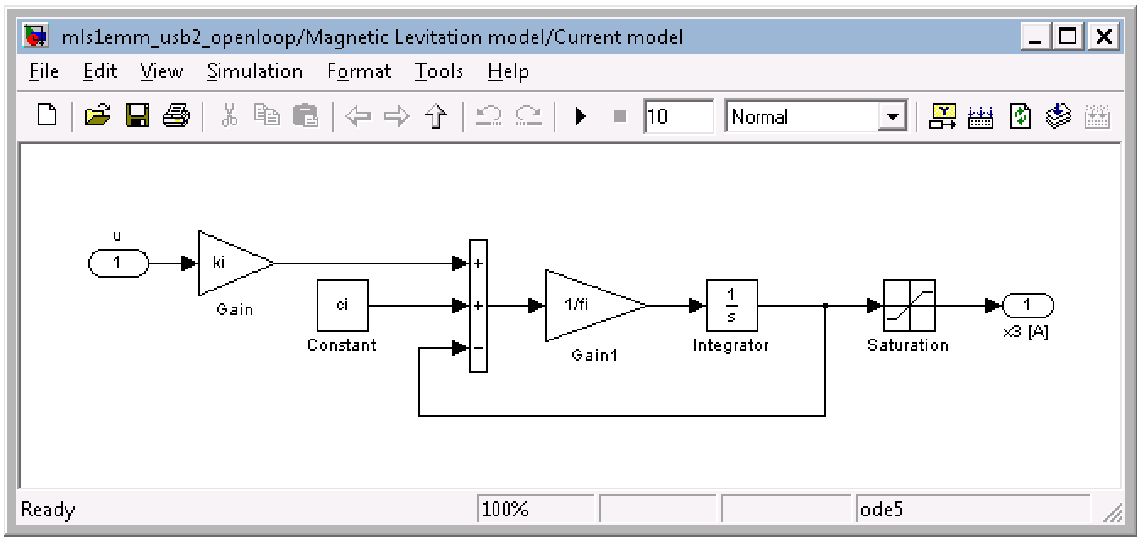

mls1emm_usb2_openloop/Magnetic Levitation model/Current model
File Edit View Simulation Format Tools Help
10
Normal
u
1
ki
Gain
ci
Constant
1/fi
Gain1
1/s
Integrator
Saturation
x3 [A]
Ready
100%
ode5

Fig. 40. Interior of the *Current model* block

The Simulink model is also equipped with the animation block. When a simulation starts the following window opens (see Fig. 41). The animation screen is updated in every sample time. All state variables: the ball position and velocity, and also the coil current are animated.

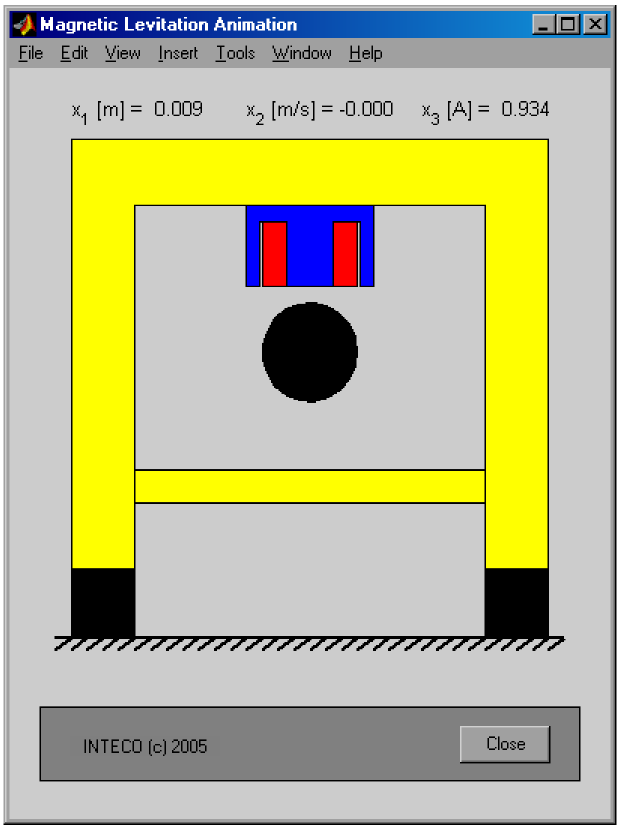

Magnetic Levitation Animation
File Edit View Insert Tools Window Help
x1 [m] = 0.009
x2 [m/s] = -0.000
x3 [A] = 0.934
INTECO (c) 2005
Close

Fig. 41. ML animation

- **Mathematical model**

Available after purchasing the system.

The parameters of the above equations are shown in the table below

Para meters	Values	U nits
m	0.0571 (big ball)	[kg]
g	9.81	[m/s^2]
F_{em}	function of x_1 and x_3	[N]
F_{emP1}	1.7521·10^{-2}	[H]
F_{emP2}	5.8231·10^{-3}	[m]
fi	0.6263 ·10^{-3}	[s]
c_i	0.0367586	[A]
k_i	0.5432595	[A/V]
x_{1MAX}	0.02	[m]

x_{3MIN}	0.03884	[A]
x_{3MAX}	2.345	[A]
u_{MIN}	0.0000	[V]
u_{MAX}	5.0000	[V]

The electromagnetic force vs. position diagram is shown in Fig. 42 and the electro-magnetic force vs. coil current diagram is shown respectively in Fig. 43.

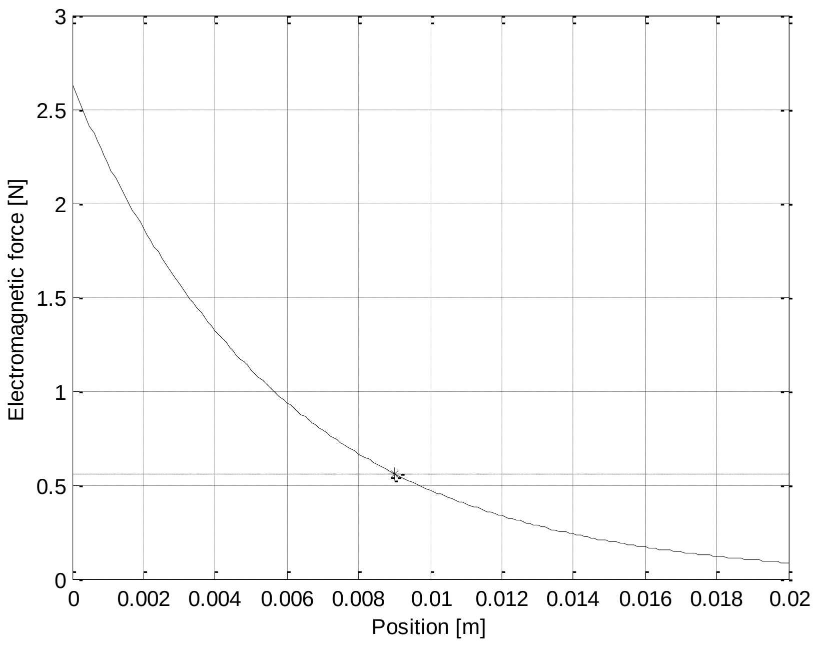


Fig. 42. Electromagnetic force vs. position. The gravity force of the larger sphere (dashed horizontal line) is crossing the curve at the 0.009 m distance from the electromagnet

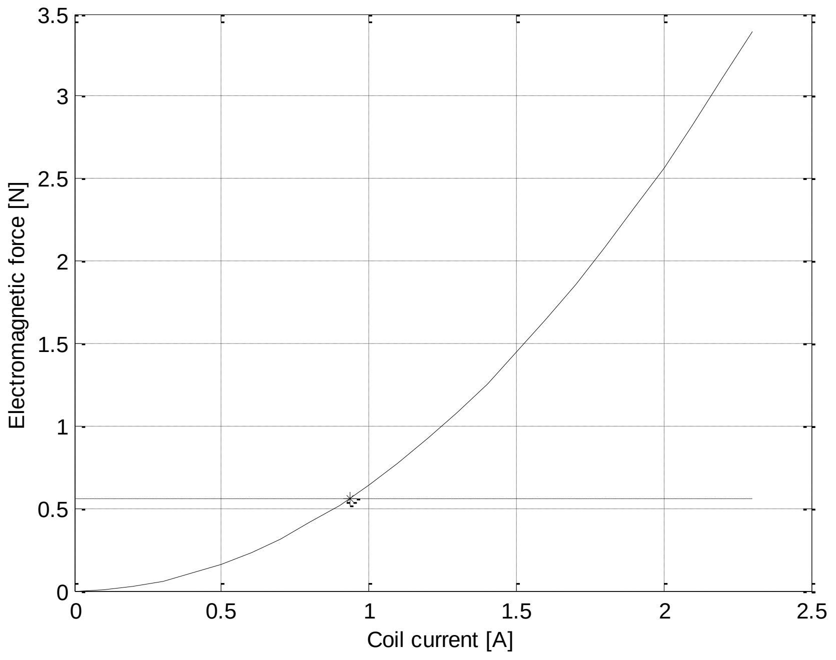


Fig. 43. Electromagnetic force vs. coil current. The gravity force of the larger sphere *m*=0.0571kg (dashed horizontal line) is crossing the curve at the 0.9345 A coil current

The electromagnetic force depends on two variables: the sphere distance from the electromagnet and the current in the electromagnetic coil. This is clearly shown in Fig. 42 and Fig. 43. We can demonstrate these dependencies in three dimensional space (see Fig. 44). The ball is stabilized at $[x_1, x_2, x_3] =$ col(9·10^{-3}, 0, 9.345·10^{-1}). It means that the ball velocity remains equal to zero. The ball is levitating kept at the 9 mm distance from the bottom of the electromagnet. The 0.9345 A current flowing through the magnetic coil is the appropriate value to balance the gravity force of the ball.

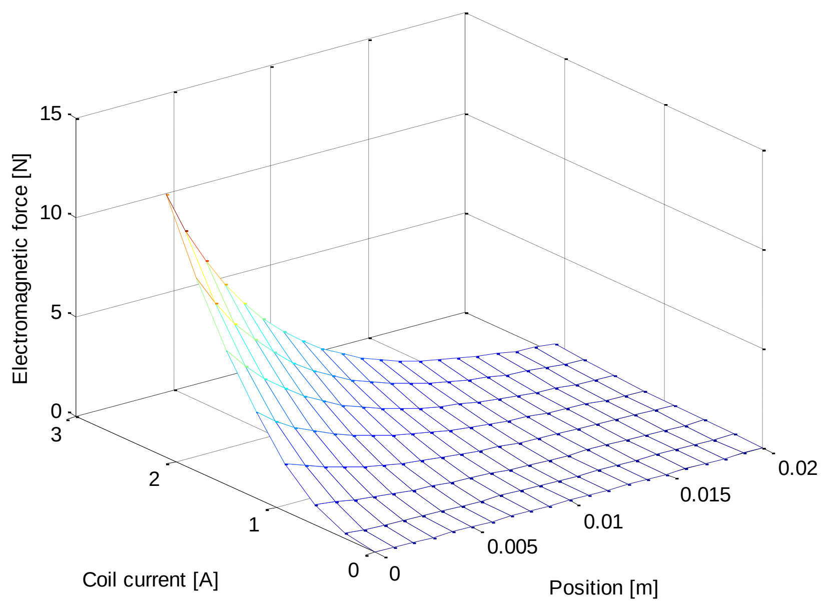


Fig. 44. Electromagnetic force vs. coil current and distance from the electromagnet.

- **Linear continuous model**

ML is a highly nonlinear model. It can be approximated in an equilibrium point by a linear model. The linear model can be described by three linear differential equations of the first order in the form:

$$\dot{x} = Ax + Bu \quad y = Cx$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ a_{2,1} & 0 & a_{2,3} \\ 0 & 0 & a_{3,3} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix}$$

The A matrix elements are expressed by the nonlinear model parameters in the following way:

Available after purchasing the system.

The C vector elements correspond to an applied controller. For example, The PID controller shown in the next subsection requires C in the form:

$$C = [1 \quad 0 \quad 0].$$

2.3.2 PID

If you click the PID button the following window opens (see Fig. 45).

The interior of the *Magnetic Levitation model* block has been shown in Fig. 39. The PID controller is built in the form:

$$u(t) = K_P \cdot e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t)$$

$$e(t) = x(t) - x_0(t)$$

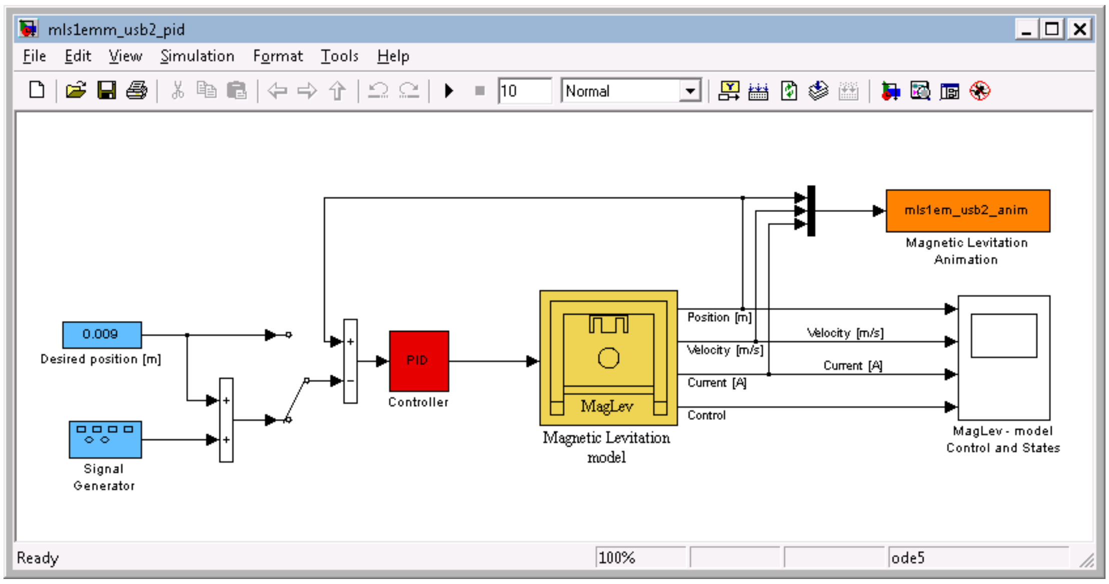


Fig. 45. PID simulation

The parameters depicted bellow are used for the PID controller.

K_P	K_I	K_D
200	1000	50

The simulated stabilization results of the 0.036kg ball are shown below.

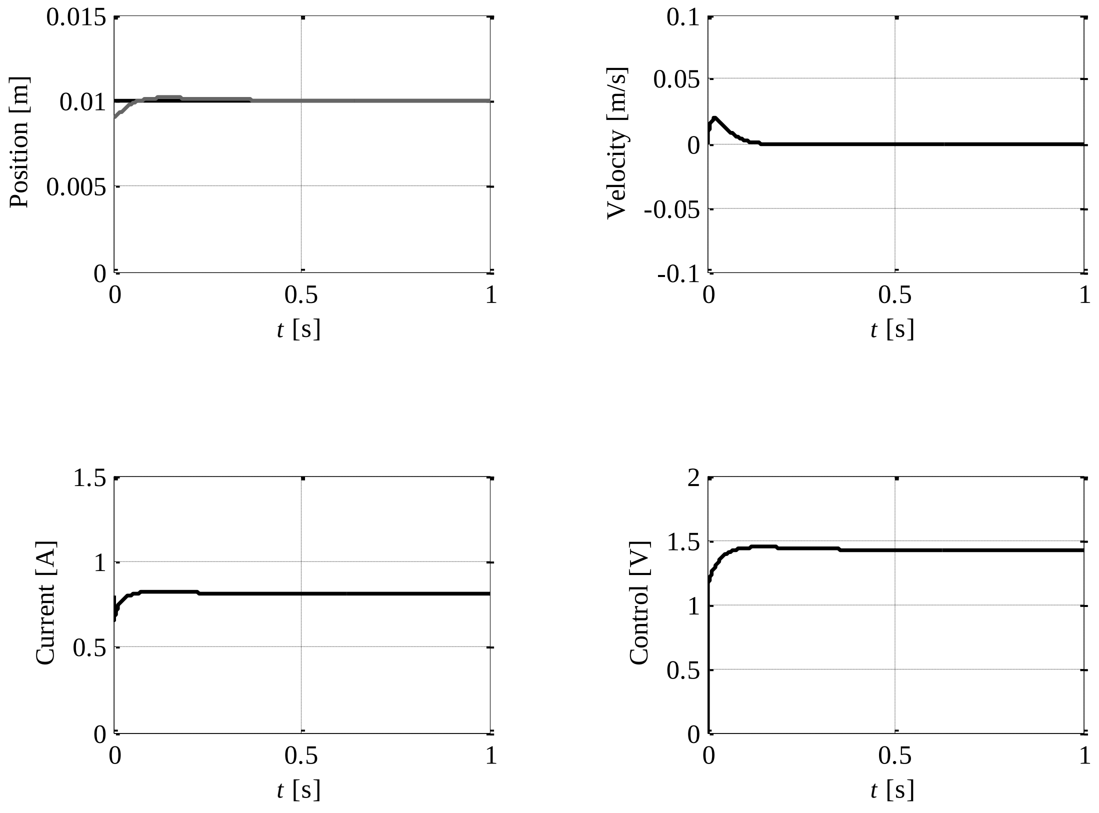
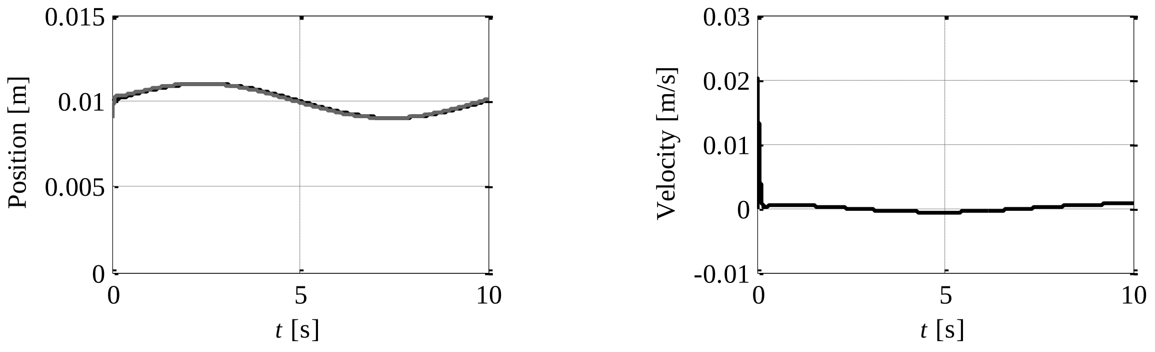


Fig. 46. PID simulation – the desired position is a constant.



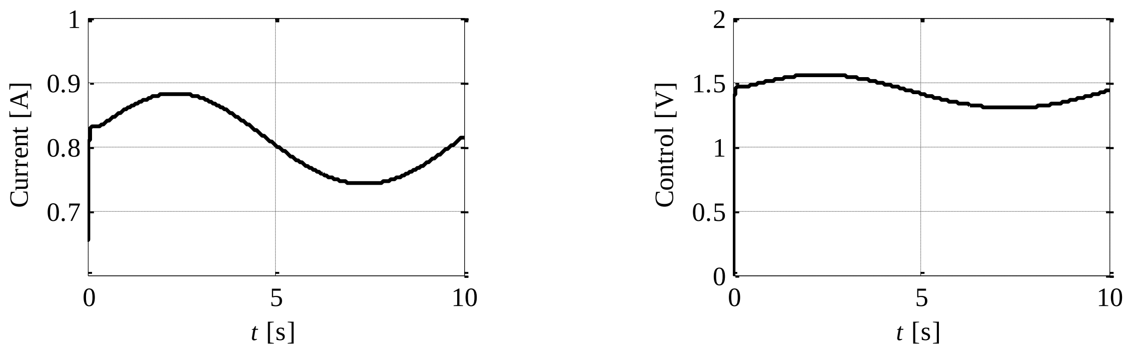


Fig. 47. PID simulation – the desired position is a sine wave.

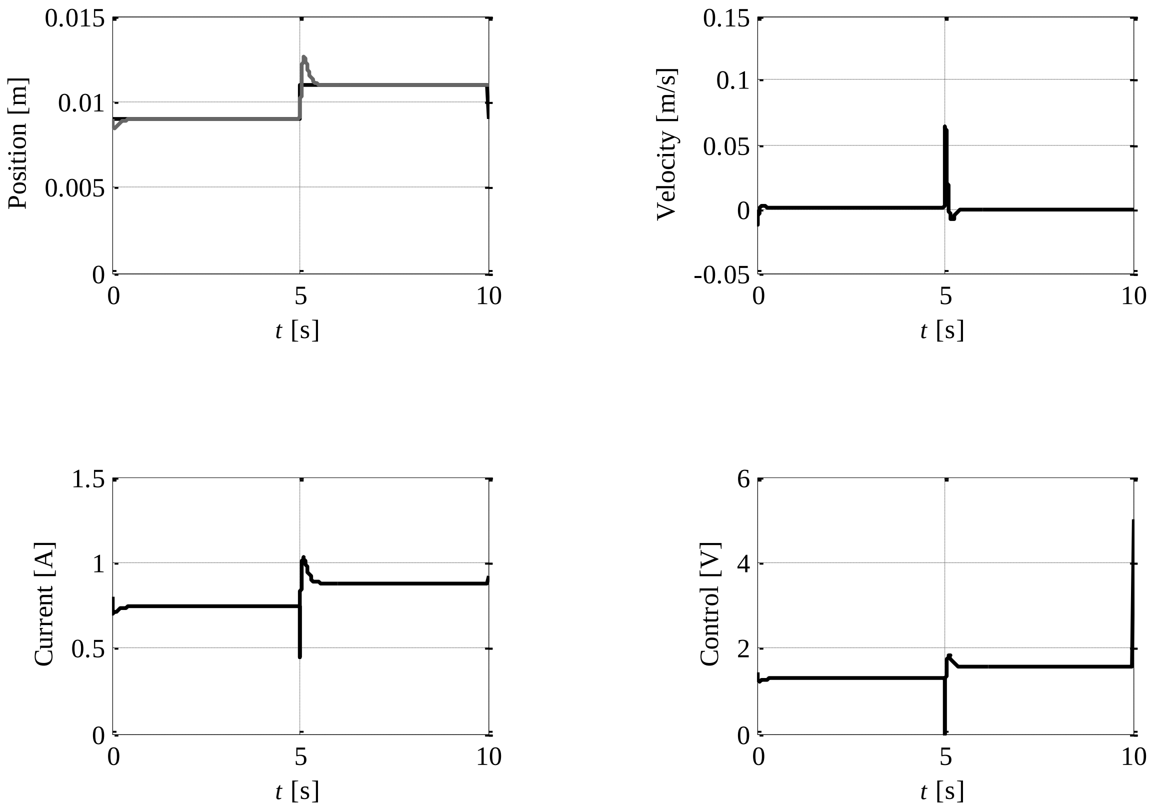


Fig. 48. PID simulation – the desired position is a square wave.

2.4 Levitation

All simulation experiments can be repeated as real-time experiments. In this way one can verify accuracy of modelling. If we double click the *levitation* button in the *MagLev Main* window the following window opens (see Fig. 49). Now, we can choose the controller we are interested in.

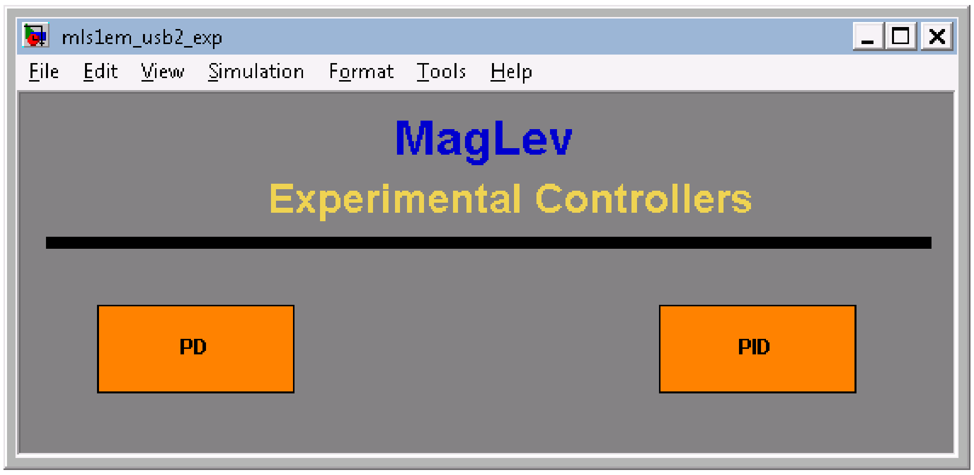


Fig. 49. Experimental controllers

To demonstrate the levitation action the PD controller has been chosen for the experiment. The results of the real-time experiment are shown in Fig. 51

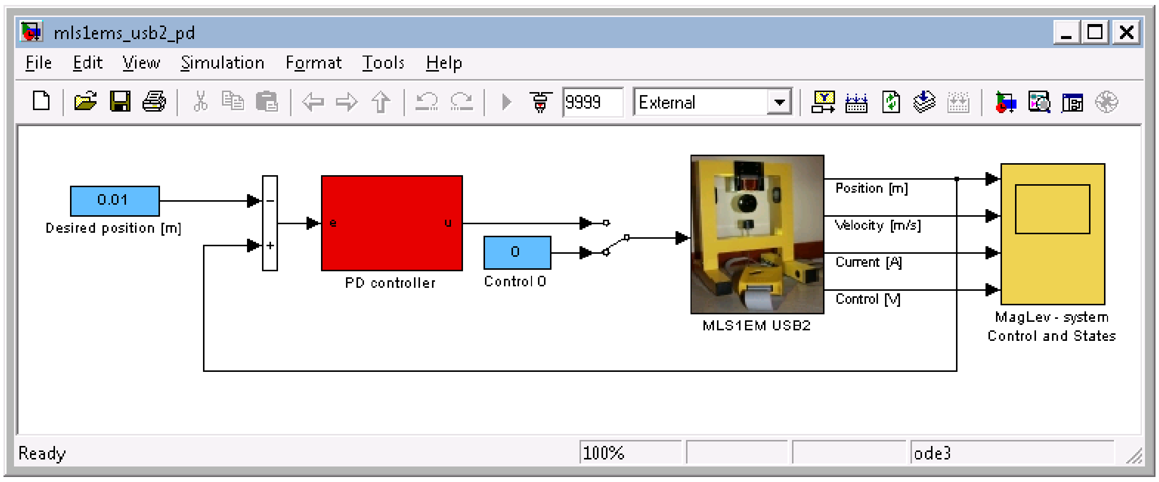


Fig. 50. PID real-time experiment.

The ball was kept manually and located in the levitation area. The ball position, ball velocity, coil current and calculated control values are plotted in time diagrams (see Fig. 51). At the beginning the ball is kept close to the electromagnet. In this case the control action is close to zero. Then the ball is placed below the stabilisation point, and the controller reacts generating the control signal. Then the ball is released and freely levitates. In the neighbourhood of the stable position. One can find that the free levitation starts at 2.5 s.

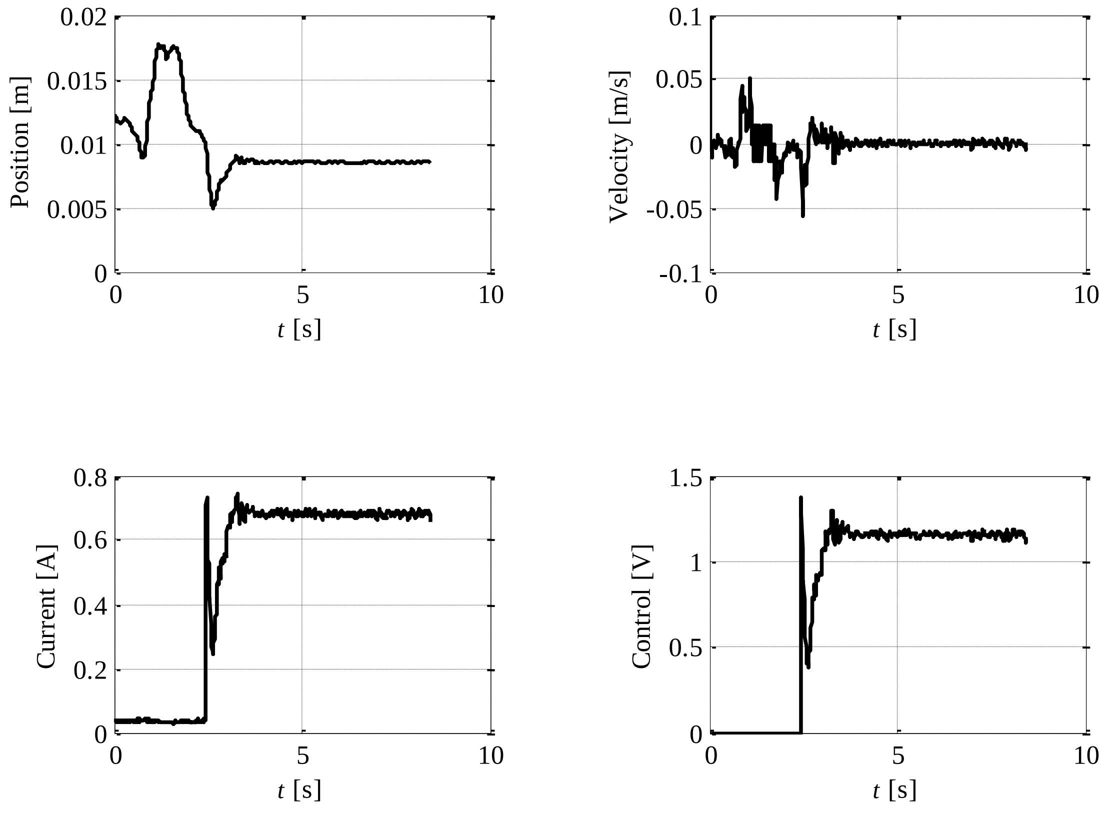


Fig. 51. PD real-time experiment. The desired position is set as a constant.

3 Real-time model in MATLAB version R2019b or newer

This section describes how to compile (build) and run a real-time model for new versions of Matlab _ R2019b or newer. The *Simulink Coder* and *RT-CON* toolboxes are used.

To build the system that operates in the real-time mode the user has to:

- create a Simulink model of the control system which consists of *Device Driver* and other blocks chosen from the Simulink library,
- build the executable file compiling model,
- start the real-time code.

The description in this section contains only the differences from the previous versions of MATLAB. So reading the previous chapter carefully is useful to understand the process of creating and running real time.

3.1 Creating a model

The simplest way to create a Simulink model of the control system is to use one of the models included in the INTECO's software. as a template. For example, click *Maglev Device Driver* model in the *Magnetic Levitation Main* window (Fig.2) The *Maglev Device Driver* Simulink model is shown in Fig. 3.1.

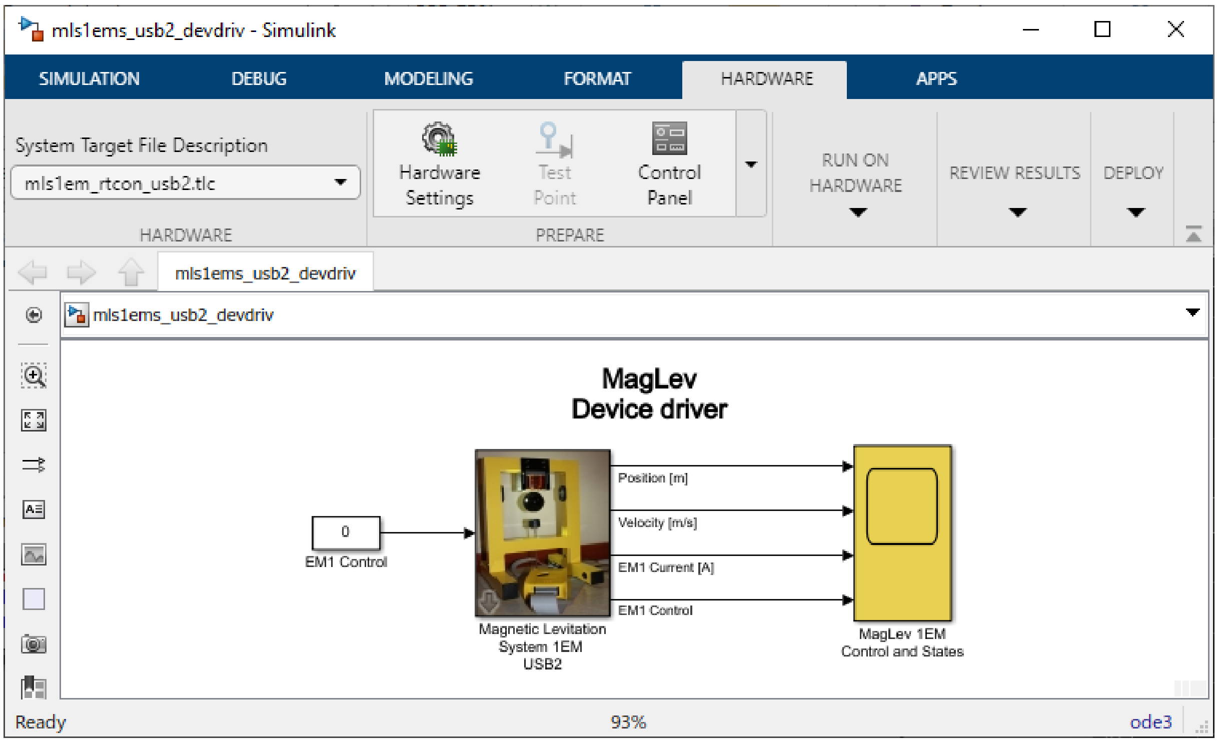


Fig. 3.1. ***Maglev Device Driver*** **real-time model**

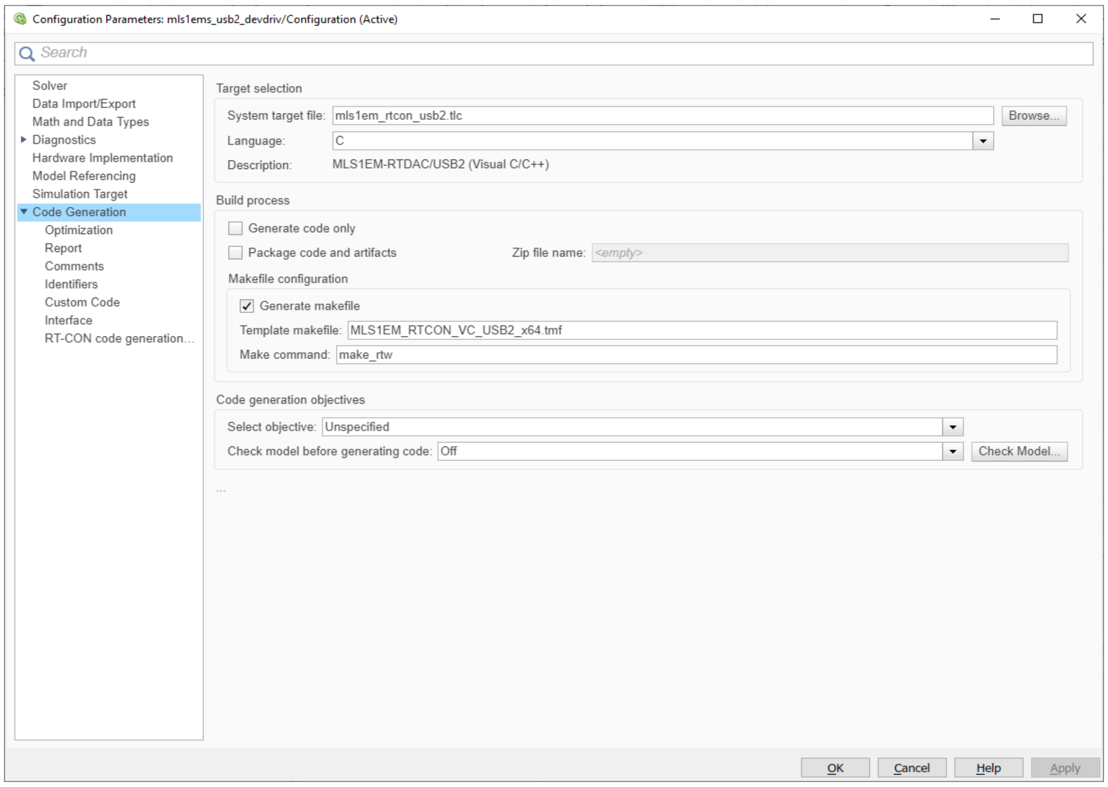
Now, you can modify the model. You get absolute freedom to develop your own controller. Remember to leave the *Magnetic Levitation System 1EM USB2* block in the window. This is necessary to work in the real-time environment.

Though it is not obligatory, we recommend you add at least one scope. You need a scope to watch how the system runs.

Creating your own model on the basis of an old example ensures that all-internal options of the model are set properly. These options are required to compiling and linking in a proper way. See at Fig. 3.2 and Fig. 3.3. To build real-time code a special files shown in *System target file* and *Template file* sections are required. These files are included to the INTECO's software.

You can apply most of the blocks from the Simulink library. However, some of them cannot be used (see Simulink Coder reference manuals).

When the Simulink model is ready, click the *Hardware Settings* option and next click the *Code Generation* button. The window presented in Fig. 3.2 opens. Select *Interface* button to see the external mode options at Fig. 3.3. The system target file name is *rtcon_tras_USB2.tlc*. It manages the code generation process.


Configuration Parameters: mls1ems_usb2_devdriv/Configuration (Active)

Search

- Solver
- Data Import/Export
- Math and Data Types
- ▸ Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- ▾ Code Generation
  - Optimization
  - Report
  - Comments
  - Identifiers
  - Custom Code
  - Interface
  - RT-CON code generation...

Target selection

System target file: mls1em_rtcon_usb2.tlc Browse...

Language: C

Description: MLS1EM-RTDAC/USB2 (Visual C/C++)

Build process

☐ Generate code only

☐ Package code and artifacts Zip file name: <empty>

Makefile configuration

☑ Generate makefile

Template makefile: MLS1EM_RTCON_VC_USB2_x64.tmf

Make command: make_rtw

Code generation objectives

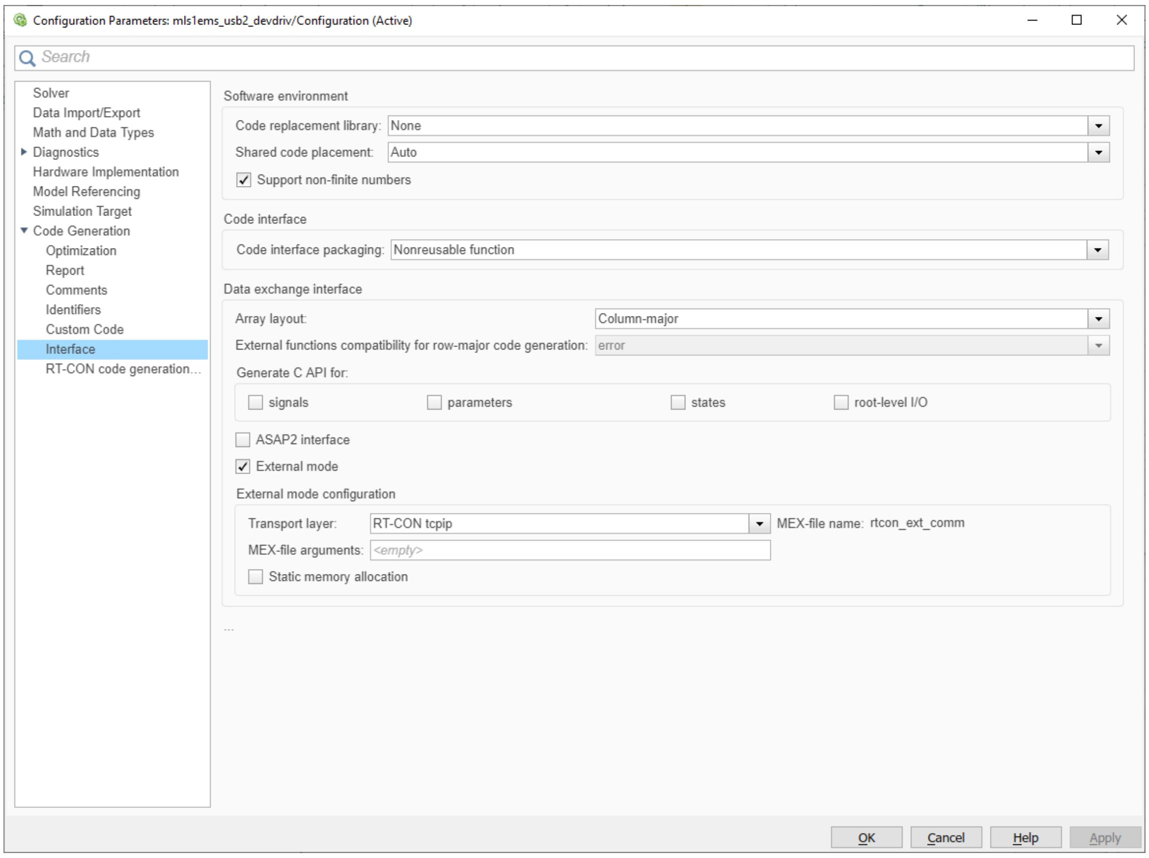
Select objective: Unspecified

Check model before generating code: Off Check Model...

OK Cancel Help Apply

Fig. 3.2 Internal code generation options

The *MLS1EM_RTCON_vc_us2bx64.tmf* template make-file is devoted to C code generation using the Visual C++ compiler.


Configuration Parameters: mls1ems_usb2_devdriv/Configuration (Active)

Software environment

Code replacement library: None

Shared code placement: Auto

[✓] Support non-finite numbers

Code interface

Code interface packaging: Nonreusable function

Data exchange interface

Array layout: Column-major

External functions compatibility for row-major code generation: error

Generate C API for: [ ] signals [ ] parameters [ ] states [ ] root-level I/O

[ ] ASAP2 interface

[✓] External mode

External mode configuration

Transport layer: RT-CON tcpip MEX-file name: rtcon_ext_comm

MEX-file arguments: <empty>

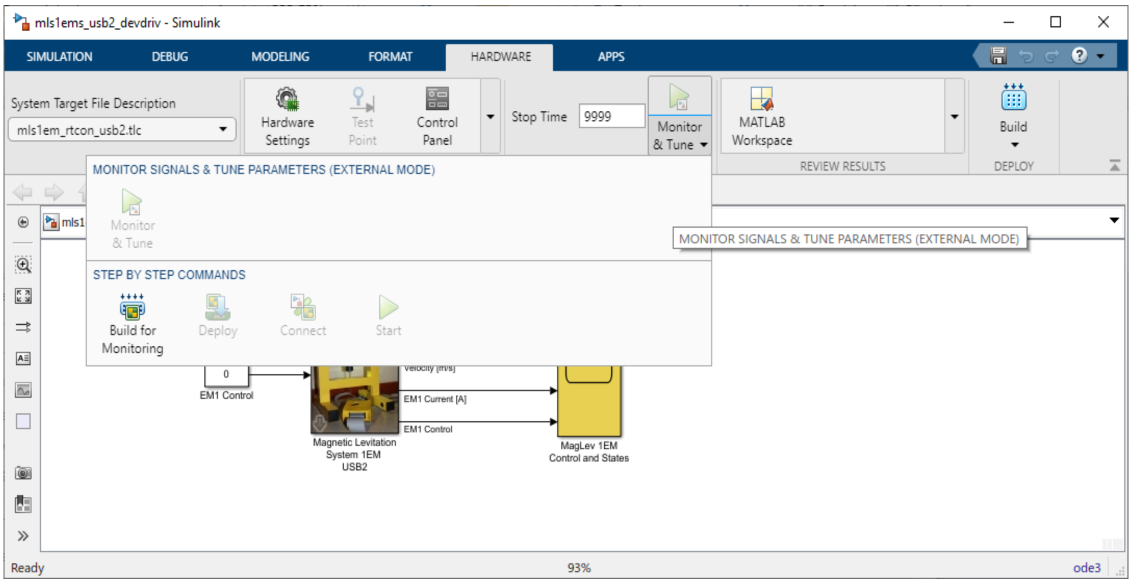
[ ] Static memory allocation

Fig. 3.3. Interface of the external mode options

3.2 Code generation and the build process

Once a model of the system has been designed the code for real-time mode can be generated, compiled, linked and downloaded into the processor.

To compile (build) model click *Monitor&Tune* button and next *Build for Monitoring* option (see in Fig. 3.4). You can also use the keyboard by clicking CTRL+b keys. You will get the same effect in this way.



mls1ems_usb2_devdriv - Simulink

System Target File Description: mls1em_rtcon_usb2.tlc

Stop Time: 9999

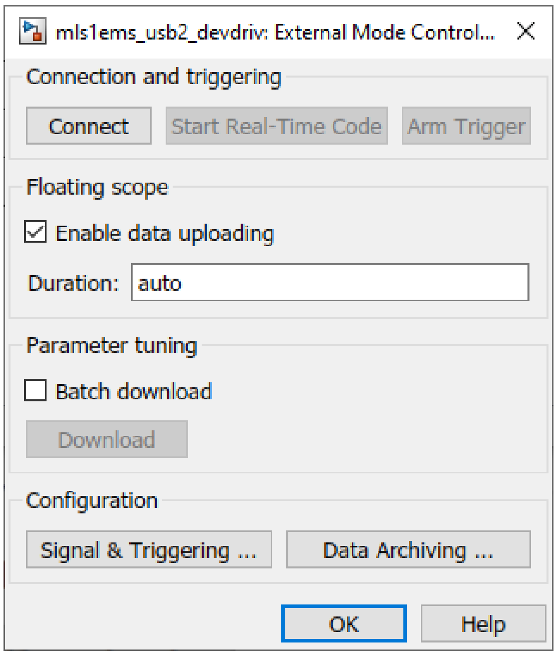
Fig. 3.4. Building model

Successful compilation and linking processes generate the following message:

### Successful completion of build procedure for model: mls1ems_usb2_devdriv
### Simulink cache artifacts for ' mls1ems_usb2_devdriv' were created in '..mls1ems_usb2_devdriv.slxc '. The build process completed successfully

Otherwise, an error massage is displayed in the *Diagnostic Viewer* window.

To run the real-time model click the *Control Panel* option and window shown in Fig. 3.5 will appear. Next click *Connect* button and real-time starts.


mls1ems_usb2_devdriv: External Mode Control...

Connection and triggering

Floating scope

☑ Enable data uploading

Duration: auto

Parameter tuning

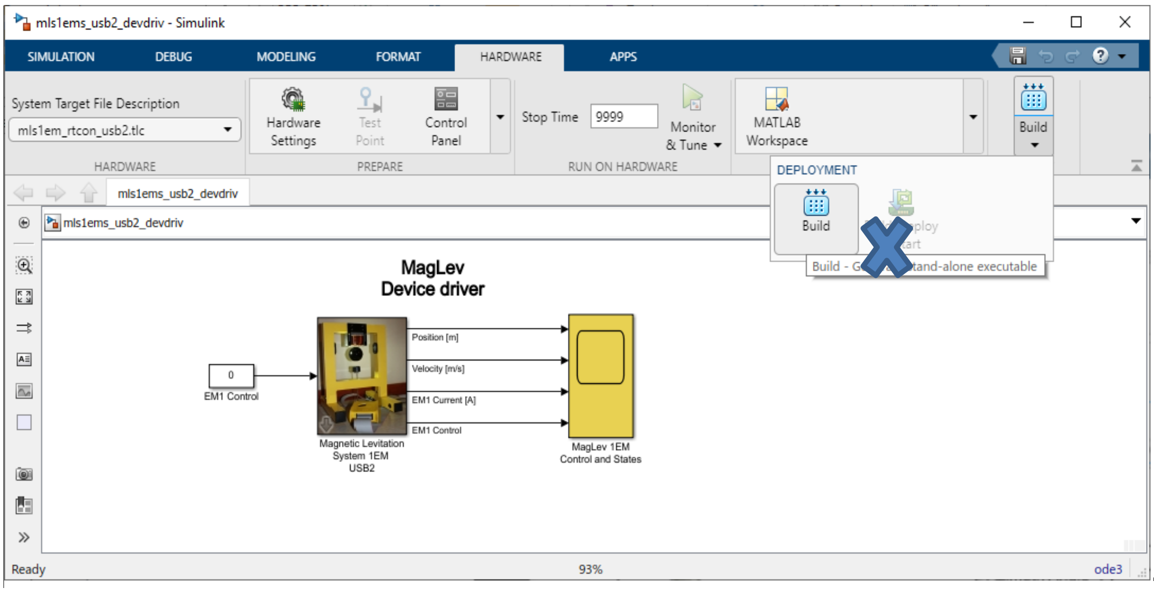
☐ Batch download

Configuration

Fig. 3.5 Connect with target

⇨ **Do not use option *Build Stand-Alone* shown in** Fig. 3.6 **for compilation .**

.


mls1ems_usb2_devdriv - Simulink

System Target File Description: mls1em_rtcon_usb2.tlc

Stop Time: 9999

Fig. 3.6 Do not use this option !!

4 References

One can find a number of references related to the Active Magnetic Levitation Systems. Here is the list of selected references when INTECO MLS systems were used.

Dragoş, C.A., Preitl, S., Precup, R.E., Petriu, E.M., *Points of View on Magnetic Levitation System Laboratory-Based Control Education*, Springer Berlin Heidelberg, Human – Computer Systems Interaction: Backgrounds and Applications 2, 2012, 978-3-642-23171-1, http://dx.doi.org/10.1007/978-3-642-23172-8_18

Piłat A. (2005). Programmable analog hardware for control systems exampled by magnetic suspension, Computer Methods and Systems, Cracow, Poland 14-16 November

Pilat A., Turnau A., Neural adapted controller learned on-line in real-time. 14 International Conference on Methods and Models in Automation and Robotics, 19-21 August, Miedzyzdroje, Poland. 2009,

http://www.ifac-papersonline.net/Detailed/41053.html

Pilat A., Testing performance and reliability of magnetic suspension controllers, 14 International Conference on Methods and Models in Automation and Robotics, 19-21 August, Miedzyzdroje, Poland. 2009,

http://www.ifac-papersonline.net/Detailed/41071.html

Pilat A., Stiffness and damping analysis for pole placement method applied to active magnetic suspension. Automatyka, ISSN 1429-3447. 2009 vol. 13 no. 1, pp. 43-54.

http://journals.bg.agh.edu.pl/AUTOMATYKA/2009-01/Auto04.pdf

Pilat A., Investigation of discrete PID controller for active magnetic suspension,

Automatyka, ISSN 1429-3447. 2010 vol 14 no. 2, pp. 181–196.

http://journals.bg.agh.edu.pl/AUTOMATYKA/2010-02/Auto02.pdf

Panuncio Cruz Francisco, Control de un sistema de levitación magnética con compensación de redes neuronales, CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS

AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL, DEPARTAMENTO DE CONTROL AUTOMÁTICO, Octubre, 2009 http://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesPCF.pdf

Pilat A., Active magnetic suspension and bearing, Modelling and simulation, eds. Giuseppe Petrone, Giuliano Cammarata. — [Vienna] : InTech Education and Publishing, 2008. ISBN 978-3-902613-25-7. pp. 453–470.

http://www.intechopen.com/books/modelling_and_simulation/active_magnetic_suspension_and_bearing

Pilat A., The programmable analog controller : static and dynamic configuration, as exemplified for active magnetic levitation. Przegląd Elektrotechniczny, Stowarzyszenie Elektryków Polskich, ISSN 0033-2097. 2012 vol. 88 no 4b, pp. 282–287

pe.org.pl/articles/2012/4b/51.pdf